CDHS

WEB-BASED

APPLICATION ARCHITECTURE

STANDARDS AND PROCESSES

Document Change Control History

Release Date	*Version*	*Release Notes*
11/03/04	3.0	New Revision Reflecting Infrastructure Modifications
02/19/04	2.2	Initial Document Release

Using this Document

This document describes the standard application development architecture for the California Department of Health Services (CDHS). This document contains details about the CDHS standard architecture, technologies, database conventions, and required presentation. This document includes the standard set of support services defined and created by the Information Technology Services Division (ITSD) to support CDHS business functions.

This document is intended to identify best practices, procedures and processes allow developers to create applications that are efficient, secure, and maintainable. This document is not intended to be a tutorial on how to write N-tier applications nor is it a replacement for professional publications or courses on how to efficient business applications.

Developing an application is a process and each application has a unique life cycle. This documentation covers the basic information regarding project life cycle management but is not intended to provide the full spectrum of project management and oversight activities that the Department must participate in while meeting State requirements.

The ITSD maintains documentation regarding server builds, processes, naming standards, software configurations on enterprise servers and other information security-specific information. This information is not to be distributed management signature of a confidentiality and non-disclosure agreement.

Audience

The audience for this document includes those sponsoring, managing and developing and hosting an application on behalf of the CDHS.

Document Quick Reference

This document contains a significant amount of information. In order to make the document more valuable it has been separated into volumes, each volume containing icon keys to identify the intended audience. In addition, it contains an abbreviated and extended Table of Contents to assist the reader in locating the information most pertinent to their assigned effort.

The volumes address the following:

⇒ **Table 1 Document Quick Reference**

Volume	**Title**	**Content**	**Audience**
N/A	Using this Document	Describes document structure, contents, audience, navigation assistance and basic introduction.	All Readers, Preferred starting point
Volume II	Core ITSD Services	Describes the operational support business functions of ITSD as a whole.	Readers interested in the range of services ITSD offers.
Volume III	Project Initiation & Management	Provides information for those that want to initiate and deploy an information technology project.	Readers interested in the process for engaging information technology use.
Volume VI	Architecture Background	Provides insight into the key considerations made in the adoption of a web-based application architecture model.	
Volume V	Application Architecture	Provides the logical requirements for web-based development for any application, the required methods for implementing an application in the CDHS network environment, and describes the required processes for utilizing the environment.	Readers interested some of the technical concepts behind the adopted architecture model.
Volume VI	Change Control	Provides infrastructure and web-based application change management information.	Readers interested in change management from a process and technical requirement perspective.

Icon Keys

This document is structured through the use of volumes to address targeted audiences. The following keys are associated with volume and section headings. The purpose of each is to indicate the type of audience that may be interested in the contents of the major topic area.

Key	Content Type	Audience
📇	Database	Application Developers, Database Administrators
🔑	Developer Specific	Application Developers
👂	Educational	Anyone
📈	Executive	Executive Staff
🌍	General	Anyone
✈	Process	Project Managers, Technical Staff
🔒	Security	Project Sponsors, Project Managers, Technical Staff
🖧	Systems Administration	Systems Administrators, Hosting Entities

ABBREVIATED TABLE OF CONTENTS

DETAILED TABLE OF CONTENTS

FIGURES
TABLE OF CONTENTS

Tables
Table of Contents

VOLUME I: INTRODUCTION

The California Department of Health Services (CDHS) is responsible for a wide range of public health issues facing California. It is responsible for disease prevention, promotion of nutritional health and quality health care for the elderly. Information technology is an essential component of maintaining and presenting information and statistics. The internet is now a critical medium for providing public health services.

The CDHS has a robust, scalable and supported infrastructure available for essential and critical business applications and reporting. This environment is approved by the Information Security Office (ISO) and is maintained as a support service to Programs by CDHS' Information Technology Services Division (ITSD). The ITSD will adopt new products, processes, services and technologies for the benefit of the enterprise. Infrastructure is available to support applications, developed on behalf of CDHS business units by State or contracted staff, through their lifecycle.

Use and adherence to information technology architecture standards and processes enables CDHS to develop and maintain business applications that are efficient, secure, and easily maintained. This manual provides both an overview and detailed description of the CDHS Information Technology Architecture Standards and Processes. Technical specifications are available by contacting the ITSD, Information Management Architecture Section.

Background

For many years the CDHS provided a general Information Technology (IT) infrastructure that allowed a wide variety of tools, methodology and support models to be implemented. While this permitted maximum flexibility for information technology development it resulted in a proliferation of development tools and database products as well as infrastructure support locations. The potential risk to business application compromise or operational cost issues is very high when applications are developed without a consistent methodology or standard tools being complicated by turnover of State and contractor support.

Recognizing the need to reduce support costs and to make more efficient use of available resources CDHS began a long-term effort to change how its physical IT infrastructure is managed and how new applications are developed. Through a collaborative process CDHS programs established standards for the purchase of computers and enterprise servers. For computers it set a standard useful life with a goal to replace a percentage of these computers each year. A similar cycle was established for enterprise servers. Each year the technical specifications for new computer and server purchases are reviewed and updated as needed. The move to the East End campus also provided an upgraded connectivity infrastructure and an opportunity to reduce the number of server support locations.

The growth in use of the internet prompts CDHS to use it as an efficient and economical means to distribute information about health issues and programs to the public and to conduct business with other healthcare organizations. The ease of access to enterprise resources connected to the internet creates a need to adhere to information technology standards so that security policies can be implemented at an operational level to protect CDHS assets. Standardization of CDHS web sites and web application development tools and methods is critical in managing a variety of project implementation and support risks.

VOLUME II: CORE INFORMATION TECHNOLOGY SERVICES

This volume describes the core essential services that the Information Technology Services Division (ITSD) provides for web hosting purposes. This does not address all services that the Division provides. You can obtain current information from the California Department of Health Services (CDHS) Intranet at http://itsd.int.dhs.ca.gov.

Standards

The following items highlight the areas of influence that the ITSD has in terms of CDHS standards and specifications:

- Project Management
- Server Procurement, Environmental Conditions, Base Agents, Build, and Configuration (this includes file, print, fax, messaging, application, web, and database servers)
- Desktop, Laptop and Mobile Computing Hardware and Software Procurement and Configuration
- Asset Management Specifications
- Database Specifications
- Web Application Architecture
- Web Site Branding & Management
- Network Connectivity
- Environment and Information Security

Project Management & Oversight

ITSD performs the following project management activities for CDHS:

⇒ **Table II Project Management & Oversight Functions**

Function	Description
Control Agency Liaison	1. Provide CDHS liaison to the Department of Finance and other control agencies 2. Review Feasibility Study Reports (FSR) and Purchase Requests for adherence to control agency requirements.
IT Strategic Planning	1. Align CDHS application development and information infrastructure with the CDHS Strategic Plan, procurement and project initiation support.

Function	**Description**
Enterprise Project Management	1. Promote service-driven partnership between IT and the Program Areas 2. Demonstrate improved project initiation and delivery success 3. Enable consistent, systematic, repeatable approach to project management 4. Improve the skills and professional capabilities of project managers 5. Ensure timely and quality IT planning that is consistent with CDHS business 6. Plans, strategies, plans, and objectives
Project Oversight	1. Ensure oversight of IT planning, acquisitions, and projects compliance with State control agency requirements

Administration

ITSD provides the services in the following administrative areas:

IT Procurement

- Provide oversight of all information technology procurements for CDHS
- Set policy and develop and maintain procedures that optimize the purchasing of IT goods and services.
- Develop policies and procedures to leverage CDHS purchasing power, provide a level of standardization, and meet mandated reporting requirements.
- Implement purchasing procedures to cover all IT equipment, products and services that are purchased by CDHS with CDHS funds or funds which CDHS has custodial oversight.
- Annually develop and manages the Department's primary Data Center Interagency Agreement (I/A) with Health and Human Services Data Center.
- Provide oversight for CMAS contracts with internal and external funding sources manage contracts with Health and Human Services Agency.

IT Training

- Provides a central training facility that can be used to provide training for CDHS programs.

Application

ITSD provides a broad range of support services for applications. The support is basically geared toward three support environments: 1) mainframe, 2) fat client and 3) thin-client.

General Support Functions

The general support functions provided for each of these application types include the following.

Architecture Design

- Design the connectivity among the variety of components in the CDHS information infrastructure.
- Research the capabilities and limitations of evolving hardware and software in the CDHS technical environment.

Change Management

The following change management areas are addressed via support teams in the ITSD:

- Desktop Management
- Server Management
- Infrastructure Management (see the Appendix, Infrastructure Change Management)
- Application Management (see the Appendix, Application Change Management)

Desktop and server management functions are available for change management support for a wide array of desktops and servers with distinctive microprocessors. Support of the desktop includes supporting Windows Operating Systems (OS) and over 6,000 office suites. Providing support for these OS' and microprocessors increased in the degree of change being imposed on the computing infrastructure. This includes solutions for developing application scripting and deploying software for the desktop, "Tier Management" for servers and desktops to accommodate operational changes to the production environment.

Infrastructure management encompasses desktops, servers and the networks. Any introduction or modification to existing standards, configurations or the introduction or removal of new nodes in the CDHS network are addressed in this change management process.

Application management is a newly developed process for the purpose of addressing the deployment and maintenance of the application hosing environment that deals more specifically with code and database changes.

Database and Data Reporting

ITSD provides support for enterprise and program specific database requirements in the following areas:

DB2 Database Administration & Business Objects Reporting

- Maintain the enterprise DB2 database located at the Health and Human Services Data Center (HHSDC).
- Set data access standards to provide a secure, reliable, high performance computing platform for CDHS program business applications that utilize this database platform.

SQL Database Administration

- Configure, install and maintain the database development, test, and production servers, utilities and application tools on the network.
- Allocate system storage and plan future storage requirements to meet program application needs.
- Create primary database storage structures (tablespaces) once application developers have designed an application.
- Control and monitor user access to the production database servers.
- Monitor and optimize the performance of the production database servers.
- Plan for backup and recovery of production database information.
- Mentor and assist application developers in the use of industry best practices for data access methodology.
- In coordination with the Information Security Office and the Production Server Hosting function, develop and maintain policies and procedures that meet information security rules and regulations.
- Maintain standard for accessing data using the data access layer.
- Participate in database risk management assessment and mitigation.

Data Reporting

- The Business Objects support function is responsible for making this enterprise reporting tool available to CDHS programs. This highly flexible tool can provide secure access over the Internet to both DB2 and SQL databases. Design database aggregations that in turn will provide ad hoc access for reporting.
- Provide report and database design assistance as well as training and mentoring for programs.
- For small applications, Microsoft's reporting services are supported.

Mainframe

ITSD supports the application environment in conjunction with the HHSDC in the following areas.

Infrastructure and Security

- Management of large mainframe projects
- Service Request (SR) processing
- Equipment inventory
- VTAM definition requests
- Mainframe printing
- VTAM Printing Subsystem (VPS)
- NetView
- Publication of departmental bulletins
- User ID administration (TSO
- CICS
- TAO-CICS e-mail
- Teale HRIS)
- RACF security
- Administration of the Master Rental Agreement (MRA) contract for both routers and 3270 equipment
- Circuit requests
- Administers the CDHS remote access accounts
- Provides assistance on planning conversions from mainframe to LAN based solutions, analyzes billing data, and performs project management

Data Guidance

- The Data Guidance function is to coordinate the operation of the enterprise's mainframe systems through scheduling, job submission/balancing, coordination with customers and output review and distribution. Processing is performed on two shifts, plus overtime on weekends.

Key Data Entry

- The Key Entry function is responsible for the transcription of information into relevant electronic media.

Development & Production Application Support

Provide design, development and support of computer applications using industry best practices. Currently these services are provided to two categories of CDHS Programs.

⇒ **Table III Application Support by Customer**

Support	***Medi-Cal Applications***	***Public Health and Administration Divisions of CDHS***
Customers	▪ CDHS Medi-Cal Program ▪ Counties ▪ Fiscal Intermediaries ▪ Federal Government ▪ Health Plans ▪ Universities ▪ Health Care Providers (Hospitals / Clinics)	▪ Women, Infants and Children Supplemental Food (WIC) Branch, Primary Care and Family Health Division ▪ WIC Local Agencies ▪ Immunization Branch, Preventive Services Division ▪ Policy and Training Branch, Licensing and Certification Division ▪ Licensing and Certification District Offices ▪ Refugee Health Branch ▪ Administration Division ▪ Personnel Branch ▪ Departmental Personnel Liaisons ▪ Federal Government ▪ Counties
Services	▪ Provide application design, development and maintenance	▪ Application Development and Maintenance Support ▪ Technical Support (DB2, VSAM and

	(ranging from minor fixes to large, multi-year development projects), production support, and ad hoc reporting. ▪ The section also provides a full-time MEDS help desk. ▪ Support for CDHS IT Applications other than Medi-Cal	ADABAS backup and recovery) ▪ Control Agency Liaison Support (helping other CDHS entities with IT-related FSRs, external contract BCPs and special project reports) ▪ Help Desk support for some systems ▪ Hardware support (ACLAIMS, WICinfo) ▪ Technical support (ACLAIMS, WICinfo)

Web

The technology trend is to provide access to information and data via the web. Support services include the dissemination and refresh of data and information via the Internet for employees, business partners and the public. The core services in this support area include the following.

Content Management

- Develop templates that can be used by programs to maintain a similar "look-and-feel" across the CDHS Internet presence, commonly referred to as "branding."
- Mentor and train program web coordinators in the use of standard web design and maintenance tools.
- Assist programs in "tagging" the information content of their websites so it can be easily retrieved by an end user content search.
- Design and maintain an organized structure for over 10,000 CDHS web pages to minimize maintenance effort by program web coordinators.
- Provide publishing mechanisms with approvals included in workflow.

Web Applications

- Develop standards for application development resulting in the dynamic retrieval of data from various sources.
- Provide an infrastructure and the methods for engaging in the utilization of the infrastructure for data management and reporting purposes
- In conjunction with other infrastructure functions, document an application development model that allows CDHS programs to build scaleable, reliable, secure and manageable applications in the appropriate security environment.
- Implement and develop standard components and tools that can be utilized by any customer using the documented application development model.
- Compile a shared library of standards and procedures, designs, and working test models along with operating procedures.
- Coordinate technical teams to research and test proposed application development methodologies and tools that may impact standard security and operating procedures.
- Provide a library of components used to provide database access for internet, intranet, and extranet applications based on the physical architecture and security requirements.

Security

Security is an integral component of securing the business of the Department. The following areas of service are provided.

⇒ **Table IV Information Security Support Functions**

Disaster Recovery and Operational Recovery Planning	▪ Implement Disaster Recovery and Operational Recovery Planning ▪ Work closely with the Health and Human Services Agency Data Center (HHSDC) on recovery strategies for CDHS information residing at HHSDC.
Information Security	▪ Oversee the implementation of an effective virus protection program, intrusion detection systems, homeland security preparedness, security audits and investigations program, and the incident response program. ▪ Provide annual training to employees and contract staff on the current CDHS Information Security policy ▪ Conduct information security annual policy reviews and make revisions as necessary. ▪ Conduct Feasibility Study Report reviews ▪ Provide security and privacy consultation, application and infrastructure assessments, and review of requests for security practice reviews.

Infrastructure

ITSD provides the core support for conducting business in the enterprise. The architectural technology base for CDHS includes basic support services as follows.

Network Connectivity

- Local and wide area network (LAN/WAN) support and cabling for CDHS and its customers.
- It installs and configures the network software and hardware (switches, hubs, routers, firewalls, patch panels), defines infrastructure standards, provides network management, and performs project management.
- It is responsible for the design, implementation, operation, troubleshooting, capacity planning, change management, and maintenance

of the complex, multi-protocol LAN/WAN environment and the network infrastructure.

Asset Management

- Tracking and reporting on IT assets (hardware and software products) that are owned (IT property with CDHS state tags) and utilized by CCDHS staff
- Deployment of an automated data collection tool (SMS) and the integration of the inventory data into the Remedy Help Desk/Asset Management system
- Preparation of a software management plan containing the information that is required to be submitted annually to the Department of Finance.

Desktop Support

- Provide Desktop Tier Management. This is the process that provides for the modification of any part of an organization's distributed desktop computer environment and that subsequently supports the acceptance, approval and implementation of those modifications to the desktop and infrastructure.

Server Hosting

- Performs business needs assessments for the CDHS server environment.
- Reviews new information regarding new OS products, upgrades, OS patches, service packs, security patches.
- Analyzes functionality and features of these products.
- Develops recommendations for which new technologies to implement.
- Engineers and tests new technologies on non-production servers.
- Develops specific procedures and processes for the implementation of a technology into the current production environment.
- Oversees implementation of technologies in production environment.
- Monitors and reviews how the new technology behaves in the environment, troubleshoots and researches issues that arise and proposes fixes to problems that arise in production.
- Interfaces with the application developers and customers to determine if business needs are being met.

Messaging

- Centralized e-mail, calendaring, public folder and faxing support and administration for over 6700 CDHS staff and contractors.
- Provides assistance at the desktop level as well as the server level.
- Provide training on messaging software usage to CDHS staff.

LAB

- Provides a test bed for evaluating technology, isolating it from the production network. See Appendix, Department of Health Services Research Center (CDHSRC) Architecture and Design for more information.

Help Desk

1. Call Receipt - Document/log all calls
2. Call Resolution or Call Routing (level 2)
3. Call Tracking until resolution
4. Call escalation
5. Hot incident Management
6. Call Closeout
7. Quality Assurance

VOLUME III: PROJECT INITIATION

This volume describes the core business process that dictates the life cycle of a project. You can obtain the most current information regarding project management on the California Department of Health Services (CDHS) intranet at http://itsd.int.dhs.ca.gov/Planning%20and%20Project%20Management/ or by contacting the Information Technology Services Division (ITSD) Project and Project Management Branch.

ITSD Services

Starting the process of building an application at CDHS is the most critical element of deploying an application. Not following the proper procedures when you start the project can result in extensive delays in project implementation. The following documentation is designed to provide our consumers with procedures for initiating projects in CDHS.

Project Management Approach

Each information technology project in the CDHS is subject to a variety of approvals and adherence to State processes and procedures for initiation, development, deployment and closure. The following summarizes the basic life cycle.

Introducing the Project

The first step in developing a business application is to prepare a Project Concept Paper. The level of detail in the concept paper should be commensurate with the degree of complexity and potential risks associated with the proposed initiative. This document is sent to the ITSD Planning and Oversight Section (POS) for processing. POS staff will review it to determine if it is complete and if so, then forward the document to the CDHS Chief Information Officer (CIO) and the Deputy Director of Administration. The CIO reviews the document for technical feasibility and the Deputy Director of Administration reviews the fiscal feasibility. After their review, the document is presented to the Chief Deputy Directors for their approval to proceed with the initiative.

After the concept is approved by the Chief Deputies, the Program completes the appropriate type of feasibility study report (please see the table below). POS provides the forms/templates for the appropriate documents. After all FSR approvals are received, then the development and procurement phase can begin.

If the program's proposed solution to meet a business need, problem, or an opportunity has a total one-time IT cost of over $30,000, then the Program first completes an IT Project Concept Paper and obtains pre-approval from the sponsoring Deputy Director.

⇒ **Table V Project Reporting Requirements Summary**

Project Criteria	Type of Document Required	Approvals Required	Follow-up Reporting Required after approvals
Estimated total costs > $1,000,000 Or Requires a budget action Or Legislatively mandated Or Others as required	Feasibility Study Report (FSR)	Program Deputy, CIO, Budget Office; Director's Office, Agency, DOF	Monthly status reports to Agency, POS Independent Project Oversight reports; PIER at end of project
Cost > $50,000 but less than $1,000,000; doesn't require budget action; not legislatively mandated;	Internal FSR (IFSR)	Program Deputy, CIO	Annual update to strategic planning documents
Cost > $30,000 but less than $50,000; doesn't require budget action; not legislatively mandated.	Abbreviated IFSR (EZ FSR)	Program Deputy, PPMB, CIO	None

The Internet to Extranet (I2E) committee will work with the Program and project oversight entities on the architectural solutions that do not exist as an existing and supported architecture. The I2E committee develops technical standards, determining how to best implement new technologies while assisting CDHS staff in developing and successfully implementing IT initiatives. The members of this committee are knowledgeable in a broad spectrum of IT processes and tools. This committee reviews the Concept Paper or the FSR to ensure that the proposed solution complies with CDHS technical standards and best practices. This group also initiates infrastructure enhancements or modifications ensuring project success. You can obtain feedback from the members of the I2E committee regarding design or project-related issues by sending an e-mail to CDHS I2Eteam@dhs.ca.gov. You may be asked to be present at the regular meetings to discuss issues and actions necessary to assist your project team in development, testing and deployment phases.

There will be several meetings that must take place in order to get your project approved by control entities and to get your project launched on time. You will need to meet with members of the following groups within the ITSD throughout the lifecycle of your project:

1. Network Infrastructure Unit (NIU) - concerning network bandwidth and connectivity issues
2. Information Security Office (ISO)- regarding any possible confidentiality and security issues

3. Server Unit (NTSS) - for server specifications if your project requires new servers
4. SQL Server Unit (SSU) - for database design, security standards, backup and disaster recovery, auditing, monitoring, capacity planning and performance tuning
5. Internet Unit (IU)- for web services (application, static, file transfer, domain name registration, security, access, templates, site use reporting, and server configuration standards)
6. Business Intelligence Unit (BI) - for business intelligence reporting services and requirements
7. Client Technology Unit (CTU) - for workstation requirements
8. Application Support Branch (ASB) - for application development services if ITSD will be developing your application
9. Planning and Project Management Branch (PPMB) - for project initiation services, project reporting services, and project management best practices
10. Information Technology Help Desk - for support services
11. Exchange Unit – for messaging services
12. Data Guidance Unit (DGU) - for application processing, and
13. Administration - to discuss fund allocation and process orders.

The I2E team has members from a majority of the technical areas noted above. Each group produces technical documents related to their area of expertise. ITSD representatives will insist that you follow the procedures noted in initiating a project prior to any commitment of resources.

Project Management Approach

The following information is extracted from a more detailed description of the CDHS standard project management approach. The complete details are available on the CDHS Intranet at http://itsd.int.dhs.ca.gov/Planning%20and%20Project%20Management/ under a heading of "CCDHS IT Project Framework".

Concept Definition Process

In the Project Concept Definition phase, the purpose is to define at a high level the objectives, benefits, and approach of the proposed IT project or opportunity. Ideally, the data gathered provides management with the information necessary to decide if the proposed project can be supported and whether further planning and a feasibility study should be performed. If the project is not supportable, further costs associated with developing the business case, feasibility study, and plans are avoided.

Activities

- Define concept
- Assess unmet needs
- Define business problem/opportunity
- Define high-level scope
- Provide rough Order of Magnitude (ROM) estimate and schedule
- Compile Project Concept Paper
- Obtain approval

Inputs

- Documented business processes
- State, court, legislative mandates
- CDHS Strategic Plan

Outputs

- Approved Project Concept Paper

Tools & Templates

- IT Project Concept Paper Approval Process
- IT PC Instructions IT PC Paper

Initiation & Approval Process

The objective of the Project Initiation and Approval phase is to develop a comprehensive business case, feasibility study, and a plan to address the IT problem or opportunity. The results of the analysis and planning are used to understand the justification, priority and resources required for the proposed project. A key output, therefore, is the project and funding/spending approvals. Since this phase may extend for a long period of time, a formal mid-phase checkpoint review with the Sponsor is conducted in order to confirm the desire to continue and complete this phase. For large projects, a Steering Committee is formed to oversee the feasibility study and the report writing.

Activities

- Develop hi-level business and technical requirements
- Perform Cost Benefit Analysis, alternative analysis
- Perform business process modeling
- Perform Feasibility Study
- Define recommended solutions
- Prepare planning estimate (EAW)
- Prepare IT Procurement Plan (ITPP)
- Checkpoint review with project Sponsor
- Develop Project Management Plan including high-level scope milestone schedule, risk assessment
- Prepare/submit BCP and or Federal funding request if required
- Obtain IFSR/FSR, BCP and /or Federal funding approval

Inputs

- Approved IT Project Concept Paper or equivalent

Outputs

- Approved IFSR/FSR (with Project Plan & high-level requirements)
- BCP/Federal funding (grant request)

Tools & Templates

- IFSR-EZ Instructions
- IFSR-EZ Form
- Feasibility Study Report Preparation Instructions (FSR)
- IFSR TEMPLATE
- FSR-SPR Review and Approval Process
- CDHS EAW Guidelines
- EAW Workbook
- Project Org Chart

Security Review Process

The Information Security Office requires the completion of a questionnaire in the project initiation process. It is designed to assist the Information Security Office in expediting the security review process. The questionnaire is to be submitted with your IFSR/FSR/SPR. The current version of the questionnaire can be obtained from:

http://itsd.int.dhs.ca.gov/planning%20and%20project%20management/Resources%20Library/9%20ISO%20security%20questionnaire%2005-25-04.doc.

Web and Database Review Process

The Internet Unit (IU) and SQL Server Unit (SSU) require completion of a questionnaire in the project design phase of the project. It is designed to assist the support units in obtaining the information necessary to capture the project details sufficiently to assist ITSD in identifying any web application hosting requirements that may be unmet if not identified early in the project lifecycle.

Refer to the Appendix, Web and Database Hosting Questionnaire for a copy of the questionnaire. The most current version of this document will be available in this document.

VOLUME IV: ARCHITECTURE BACKGROUND

This volume addresses the various elements that CDHS security and technical teams took into consideration in developing this standard. It is important to note that the details of these considerations are partially influenced by the choice to support Microsoft technologies as a standard platform for the CDHS web-based application architecture. Thick-client or Windows Forms based applications may not be consistent with all aspects of this architecture. Architectures will be considered in the feasibility and design phases of CDHS projects.

This volume is designed for the infrastructure and application development teams that will engage in deployment of applications for CDHS.

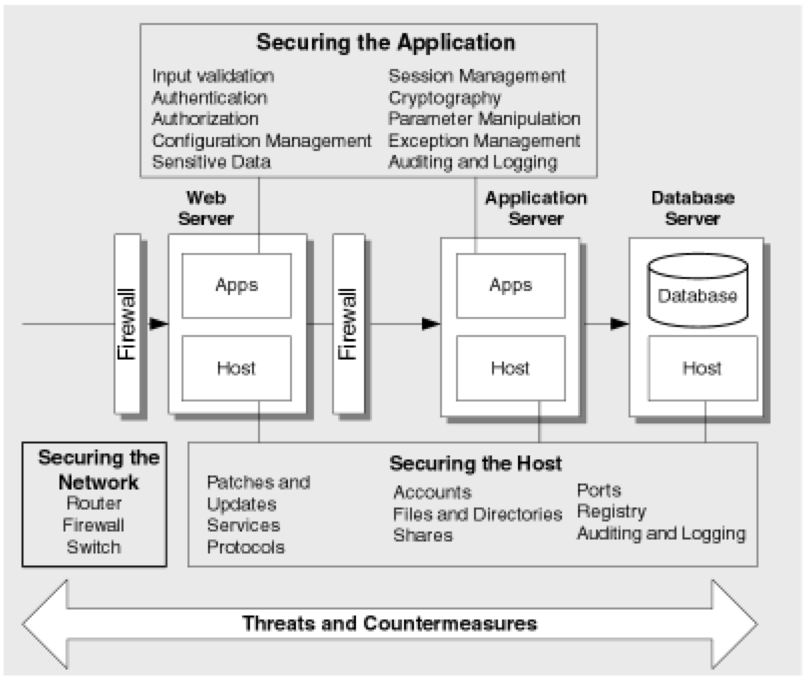
Overview

CDHS has created an environment that allows for web based applications to be hosted in a shared, secure environment. To accomplish this task a number of configuration and environmental pieces are implemented.

The Department supports n-tier architecture. There are many benefits of implementing an n-tier model from performance and security perspectives. An n-tier model allows the flexibility of increasing servers in any zone to improve performance. It also gives developers the ability to develop code for different tiers at the same time allowing team development to take place.

Security in today's enterprise computing environment has become one of the major architecture influences. CDHS implements multiple levels of security to protect its data leveraging the ability to provide isolation between tiers as is consistent with the industry approach as depicted in the figure below.

⇒ **Figure 1 Securing Web Applications**



One major security enhancement that CDHS is implementing is the ability to take advantage of Operating System (OS) security improvements provided by deploying Windows servers in an Active Directory environment. Applications and servers can be configured to use Integrated Windows Security allowing the OS to manage the transmission of user credentials in a very secure manner. ASP.NET applications can take advantage of role based security and a trusted sub-system model. The ability to create security groups at the network level by creating Active Directory global groups within organizational units provides enhanced security and ease of maintenance. By implementing a trusted sub-system model performance enhancements are achieved by using connection pooling between the application and data layers. Security is greatly improved at the data layer by not allowing the end users to have access to the data directly.

Although an n-tier model may appear to be more complicated the benefits far outweigh the drawbacks. The following material is designed to describe the major elements of the CDHS architecture considerations and the logic behind the selection of adopted methods.

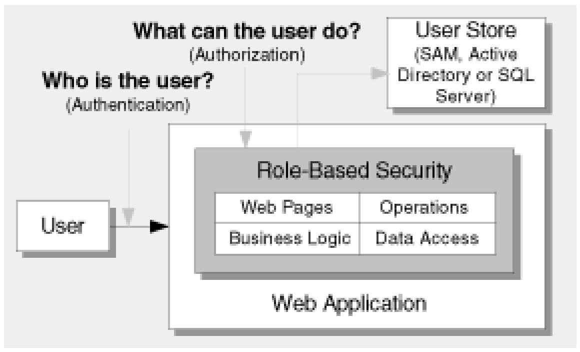
User vs. Code Security

User security and code security are two complementary forms of security that are available to .NET Framework applications. User security answers the questions, "Who is the user and what can the user do?" while code security answers the questions "Where is the code from, who wrote the code, and what can the code do?" Code security involves authorizing the application's (not the user's) access to system-level resources, including the file system, registry, network, directory services, and databases. In this case, it does not matter who the end user is, or which user account runs the code, but it does matter what the code is and is not allowed to do.

The .NET Framework user security implementation is called *role-based security*. The code security implementation is called *code access security*.

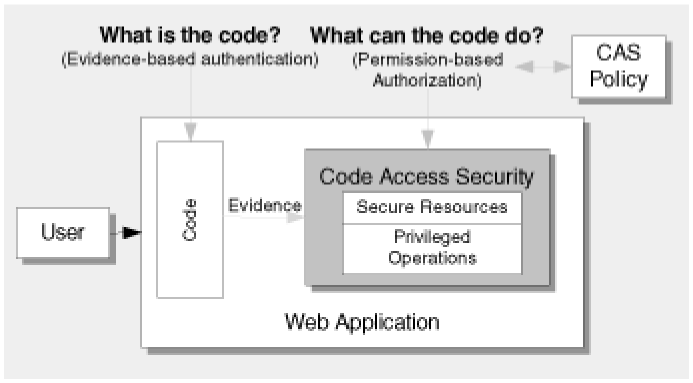
The figure below shows a logical view of how user security is typically used in a web application to restrict user access to web pages, business logic, operations, and data access.

⇒ **Figure 2 Logical View of User (Role-Based) Security)**



The figure below shows a logical view of how code access security is used in a web application to constrain the application's access to system resources, resources owned by other applications, and privileged operations, such as calling unmanaged code.

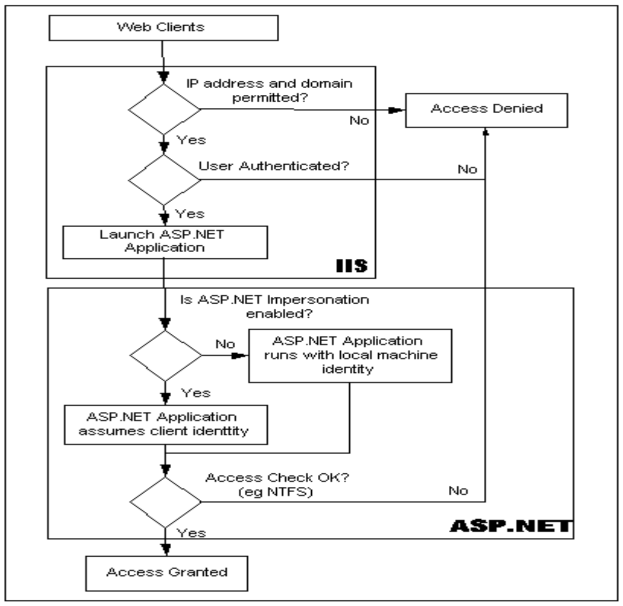
⇒ **Figure 3 Logical View of Code-Based Security**



Relationship between IIS and .NET Web Applications

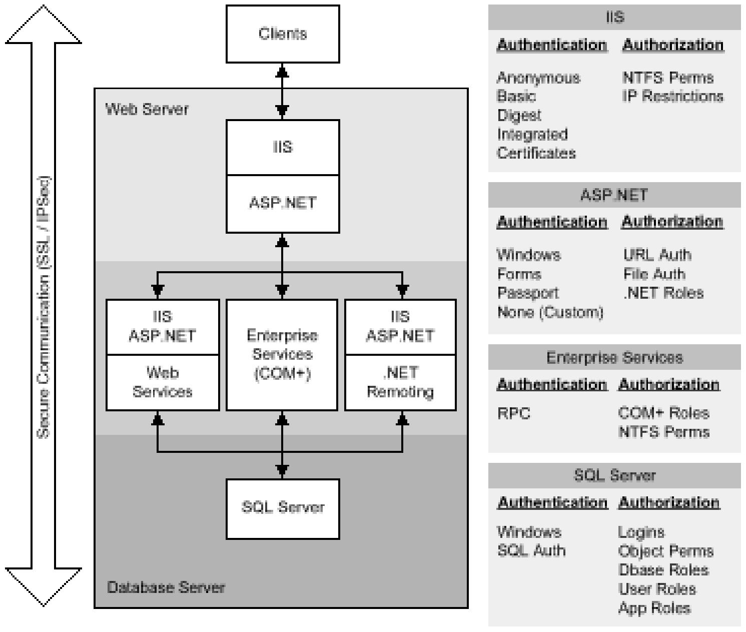
ASP.NET application security configuration and IIS security configuration are completely independent and can be used independently or in conjunction with each other. IIS maintains security related configuration settings in the IIS metabase. ASP.NET maintains security (and other) configuration settings in XML configuration files. While this generally simplifies the deployment of your application from a security standpoint, the security model applied to your application necessitates the correct configuration of both the IIS metabase and your ASP.NET application via its configuration file (Web.config). The following figure illustrates this concept.

⇒ **Figure 4 IIS/ASP.NET Relationship**


Web Clients
IP address and domain permitted?
No
Access Denied
Yes
User Authenticated?
No
Yes
Launch ASP.NET Application
IIS
Is ASP.NET Impersonation enabled?
No
ASP.NET Application runs with local machine identity
Yes
ASP.NET Application assumes client identtity
Access Check OK? (eg NTFS)
No
ASP.NET
Yes
Access Granted

As the figure depicts, there are two essential items that must be addressed in granting access to users of an application, authentication and authorization. As further demonstrated in the figure below and subsequently described, basic authentication and authorization methods are available in the main application layers.

⇒ **Figure 5 Authentication and Authorization Methods in Main Application Layers**



IIS	
Authentication	**Authorization**
Anonymous	NTFS Perms
Basic	IP Restrictions
Digest	
Integrated	
Certificates	

ASP.NET	
Authentication	**Authorization**
Windows	URL Auth
Forms	File Auth
Passport	.NET Roles
None (Custom)	

Enterprise Services	
Authentication	**Authorization**
RPC	COM+ Roles
	NTFS Perms

SQL Server	
Authentication	**Authorization**
Windows	Logins
SQL Auth	Object Perms
	Dbase Roles
	User Roles
	App Roles

Authentication

Authentication is the process of the user proving they are who they say they are. Web applications may utilize a combination of ASP.NET, Internet Information Server (IIS) authentication, Enterprise Services, and SQL authentication. When IIS authentication is complete, ASP.NET uses the authenticated identity to authorize access.

IIS Authentication

The following discusses the various authentication methods that IIS supports.

Basic: Used for non-secure identification of clients, as the user name and password are sent in base64-encoded strings in plain text. Passwords and user names are encoded, but not encrypted, in this type of authentication. A determined, malicious computer user equipped with a network-monitoring tool can intercept user names and passwords.

Basic over SSL: Used for secure identification of clients in Internet scenarios. The user name and password are sent over the network using Secure Sockets Layer (SSL) encryption, rather than plain text. This is works for Internet scenarios. SSL degrades performance.

Digest: Used for secure identification of clients in Internet scenarios but available only on domain controllers in Windows 2000, but is not the case in Windows 2003 Digest or Advanced Digest. This requires the use of domain accounts. Uses hashing to transmit client credentials in a secure manner so the password is not transmitted in clear text. In addition, Digest authentication can work through proxy servers. However, it is not widely supported on other platforms.

Integrated Windows Authentication: Uses NTLM or Kerberos with a cryptographic exchange with the user's Microsoft Internet Explorer web browser.

Anonymous: Anonymous access requires no authentication whatsoever. Anonymous access is used for web sites when you want unauthenticated users to be able to access the information provided by the web sites and applications

ASP.NET Authentication

The following discusses the various authentication methods that ASP.NET supports.

Windows Authentication: ASP.NET uses Windows authentication in conjunction with IIS authentication. Authentication is performed by IIS. When IIS authentication is complete, ASP.NET uses the authenticated identity to authorize access.

Forms: Forms authentication generally refers to a system in which unauthenticated requests are redirected to an HTML form, using HTTP client-side redirection. Forms authentication is a good choice if your application needs to collect its own user credentials at logon time through HTML forms. The user provides credentials and submits the form. If the application authenticates the request, the system issues a form that contains the credentials or a key for reacquiring the identity. Subsequent requests are issued with the form in the request headers. They are authenticated and authorized by an ASP.NET handler using whatever validation method the application specifies.

Passport Authentication: Passport authentication is a centralized authentication service provided by Microsoft that offers a single logon and core profile services for member sites. This benefits the user because it is not necessary to log on to access new protected resources or sites.

SQL Authentication

Windows Authentication: User's network security attributes are established at network login time and are validated by a Windows domain controller or a member or stand alone server via the local SAM. When a network user tries to connect, SQL Server uses Windows-based facilities to determine the validated network user name. SQL Server then verifies that the person is who they say they are, and then permits or denies login access based on that network user name alone, without requiring a separate login name and password.

SQL Server Authentication: When a user connects with a specified login name and password from a non-trusted connection, SQL Server performs the authentication itself by checking to see if a SQL Server login account has been set up and if the specified password matches the one previously recorded. If SQL Server does not have a login account set, authentication fails and the user receives an error message. This occurs by defining how authentication will occur for the provider.

Authentication Decision Points

The following is a list of decision points typically used by organizations in adopting authentication methods:

- Do users have to log in? Is a user name and password required to access the site or service?
- Is personalization required? Will the site provide personalized content, without requiring the users to log on?
- User account storage? Are user accounts stored in Windows NT domain accounts, Active Directory, or are they stored in some other data store, for example a relational database, an alternate LDAP (Lightweight Directory Access Protocol) directory service, or an XML file?
- Is single sign-on or seamless logon required? Do you want the users to log on from a logon page, or do you need authentication to happen automatically? For example, you may require automatic authentication for an automated Business-to-Business (B2B) transaction.

- Do you need to make the system extremely hard for hackers to steal user names and passwords over the network? This decision is typically made based on the sensitivity of the data available on the site.
- Will the application run on the Internet? Will the application be behind a firewall, where users are not authenticated to a domain, or will the application be intranet-based where the end users may already be authenticated to a domain?
- Do you need to delegate security context? Do you need business components to run with the caller's identity? If so, impersonation is required. Furthermore, if you need to access system resources such as message queues, databases, or file systems on remote computers, delegate-level impersonation will be required.
- Are servers and clients running the same operating system, such as Windows 2000/2003? Are you running a homogeneous environment of computers?

Role-Based Authorization

Authorization is the process of determining whether the proven identity is allowed to access a specific resource. Once authentication and authorization occur, a user has access to the authorized resources. For the user to perform any action they must have the proper authorization.

There are two main authorization models in a .NET environment. These two models are role based and resource based. CDHS took into consideration flexibility and scalability when determining that the role-based model is most appropriate for the infrastructure.

Role Based

Access to operations (typically methods) is secured based on the role membership of the caller. Roles are used to partition your application's user base into sets of users that share the same security privileges within the application; for example, Senior Managers, Managers and Employees .Users are mapped to roles and if the user is authorized to perform the requested operation, the application uses fixed identities with which to access resources. These identities are trusted by the respective resource managers (for example, databases, the file system and so on).

.NET Framework role-based security is a key technology that is used to authorize a user's actions in an application. Roles are often used to enforce business rules. For example, a financial application might allow only managers to perform monetary transfers that exceed a particular threshold.
Role-based security consists of the following elements:

- Principals and identities
- PrincipalPermission objects
- Role-based security checks
- URL authorization

Resource Based

The resource-based approach to authorization relies on Windows ACLs and the underlying access control mechanics of the operating system. The application impersonates the caller and leaves it to the operating system in conjunction with specific resource managers (the file system, databases, and so on) to perform access checks.

Methods for Achieving Authorization

In the CDHS environment two main methods of authorization applied as supportable authorization methods:

1. Trusted Sub-System
2. Impersonation/Delegation

The trusted sub-system model is generally used for larger applications where performance is a major issue. The impersonation/delegation model is used for smaller applications that might not need the performance gains of the first model. In general, the impersonation/delegation model is less complicated to implement. However, the need to address scalability in the CDHS environment is imperative.

The recommended and common pattern for role-based authorization is:

- Authenticate users within your front-end web application.
- Map users to roles via active directory.
- Authorize access to operations (not directly to resources) based on role membership.
- Access the necessary back-end resources (required to support the requested and authorized operations) by using fixed service identities.

The back-end resource managers (for example, databases) *trust* the application to authorize callers and are willing to grant permissions to the trusted service identity or identities. For example, a database administrator may grant access permissions exclusively to a specific HR application (but not to individual users).

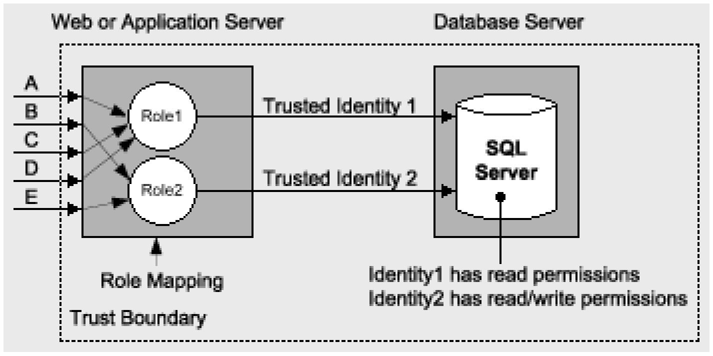
Trusted Sub-System Model

In a Trusted Subsystem Model the original identity of the user is checked at the IIS/ASP.Net gate, mapped to a role in and through active directory, then authorized based on role membership of the user. System resources for the application are then authorized at the application or role level.

A Trusted Subsystem Model controls access between the application and data tiers. The security context of the original caller does not flow through the service at the operating system level, although the application may choose to flow the original caller's identity at the application level. It may need to do so to support back-end auditing requirements, or to support per-user data access and authorization. The database trusts the application tier to authorize users and allow only authorized users to access the database using a trusted identity.

This model uses fixed accounts to access the resource being requested. When the user first logs into the application they are mapped to a role. When the user requests a resource, such as SQL Server, the role is checked and the resource is accessed by the fixed account mapped to the user's role. The image below shows an application using a multi-account setup.

⇒ **Figure 6 Trusted Sub-System Model for Authorization Based on Roles**



Advantages of the Trusted Sub-System Model

1. Scalability: The trusted subsystem model supports connection pooling, an essential requirement for application scalability. Connection pooling allows multiple clients to reuse available, pooled connections. It works with this model because all back-end resource access uses the security context of the service account, regardless of the caller's identity.
2. Minimizes Back-End ACL Management: Only a service account accesses back-end resources (for example, databases). ACLs are configured against this single identity.

3. Users Can't Access Data Directly: In the trusted-subsystem model, only the middle-tier service account is granted access to the back-end resources. As a result, users cannot directly access back-end data without going through the application (and being subjected to application authorization).

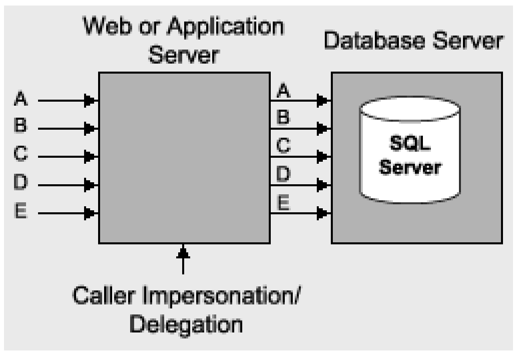
Disadvantages of the Trusted Sub-System Model

1. Auditing: To perform auditing at the back end, you can explicitly pass (at the application level) the identity of the original caller to the back end, and have the auditing performed there. You must trust the middle-tier and have a potential repudiation risk. Alternatively, you can generate an audit trail in the middle tier and then correlate it with back-end audit trails (for this you must ensure that the server clocks are synchronized).
2. Increased Risk from Server Compromise: In the trusted-subsystem model, the middle-tier service is granted broad access to back-end resources. As a result, a compromised middle-tier service potentially makes it easier for an attacker to gain broad access to back-end resources.

Impersonation / Delegation

With this model, a service or component (usually somewhere within the logical business services layer) impersonates the client's identity (using operating system-level impersonation) before it accesses the next downstream service. If the next service in line is on the same computer, impersonation is sufficient. Delegation is required if the downstream service is located on a remote computer as shown in the image below.

⇒ **Figure 7 Impersonation/Delegation**



The suggested approach when using this model is to create a Windows group or groups. For example, if we were to look at the reporting scenario again the user would log into the application in the same manner as with the Trusted Sub-system. A Windows group is created to house all users that need access to the reports. The Windows group is added to SQL Server and given read access to the report data. When the user accesses the reports their credentials are used rather than a fixed account as in the Trusted Sub-system model.

Advantages of the Impersonation / Delegation

1. The primary advantage of the impersonation / delegation model is auditing (close to the data). Auditing allows administrators to track which users have attempted to access specific resources. Generally auditing is considered most authoritative if the audits are generated at the precise time of resource access and by the same routines that access the resource.
2. The impersonation / delegation model supports this by maintaining the user's security context for downstream resource access. This allows the back-end system to authoritatively log the user and the requested access.
3. Code that accesses data cannot specify the security context therefore, a system that is compromised cannot access data based upon a different account and only based upon the current security context of the process.

Disadvantages of the Impersonation / Delegation

1. Technology challenges: Most security service providers don't support delegation, Kerberos is the notable exception. Processes that perform impersonation require higher privileges (specifically they act *as part of the operating system* privilege). (This restriction applies to Windows 2000 Server and will not apply to Windows 2003 Server).
2. Scalability: The impersonation / delegation model means that you cannot effectively use database connection pooling, because database access is performed by using connections that are tied to the individual security contexts of the original callers. This significantly limits the application's ability to scale to large numbers of users.
3. Administration Effort: ACLs on back-end resources need to be maintained in such a way that each user is granted the appropriate level of access. When the number of back-end resources increases (and the number of users increases), a significant administration effort is required to manage ACLs.

Authorization Decision Points

The following is a list of decision points typically used by organizations in adopting authentication methods:

- Where should I perform authorization and what mechanisms should I use?
- What authentication mechanism should I use?

- Should I use Active Directory® directory service for authentication or should I validate credentials against a custom data store?
- What are the implications and design considerations for heterogeneous and homogenous platforms?
- How should I represent users who do not use the Microsoft® Windows® operating system within my application?
- How should I flow user identity throughout the tiers of my application? When should I use operating system level impersonation/delegation?

Code Access Security

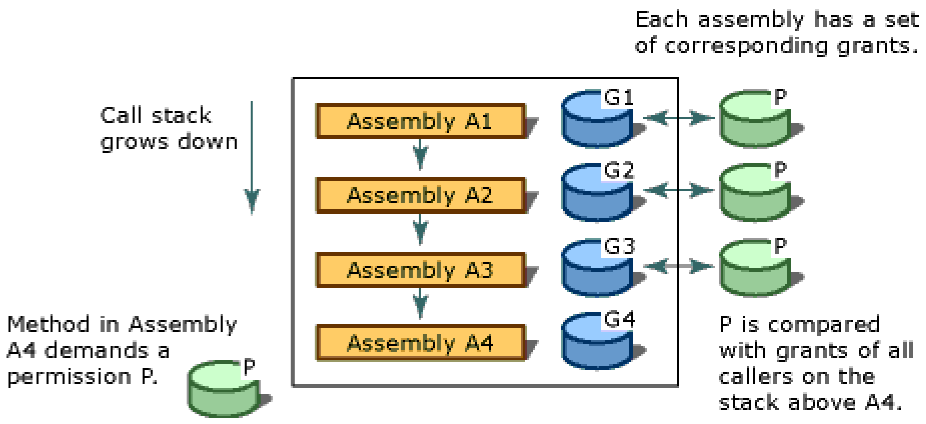
Code Access Security (CAS) is the process of allowing the administrator to control the level of permission granted to a given assembly based on the origin of the identity and origin of the assembly. CAS provides very granular control over which resources the code has access to. CAS gives the administrator the ability to isolate applications from one another. By controlling the security based on the code and not the user executing the code enhances security a great deal.

Code access security is a mechanism that helps limit the access code has to protected resources and operations. In the .NET Framework, code access security performs the following functions:

- Defines permissions and permission sets that represent the right to access various system resources.
- Enables administrators to configure security policy by associating sets of permissions with groups of code (code groups).
- Enables code to request the permissions it requires in order to run, as well as the permissions that would be useful to have, and specifies which permissions the code must never have.
- Grants permissions to each assembly that is loaded, based on the permissions requested by the code and on the operations permitted by security policy.
- Enables code to demand that its callers have specific permissions.
- Enables code to demand that its callers possess a digital signature, thus allowing only callers from a particular organization or site to call the protected code.
- Enforces restrictions on code at run time by comparing the granted permissions of every caller on the call stack to the permissions that callers must have.
- To determine whether code is authorized to access a resource or perform an operation, the runtime's security system walks the call stack, comparing

the granted permissions of each caller to the permission being demanded. If any caller in the call stack does not have the demanded permission, a security exception is thrown and access is refused. The stack walk is designed to help prevent luring attacks, in which less-trusted code calls highly trusted code and uses it to perform unauthorized actions. Demanding permissions of all callers at run time affects performance, but it is essential to help protect code from luring attacks by less-trusted code. To optimize performance, you can have your code perform fewer stack walks; however, you must be sure that you do not expose a security weakness whenever you do this. The following figure illustrates the stack walk that results when a method in Assembly A4 demands that its callers have permission P.

⇒ **Figure 8 Security Stack Walk**


Each assembly has a set of corresponding grants.
Call stack grows down
Assembly A1
Assembly A2
Assembly A3
Assembly A4
G1
G2
G3
G4
P
P
P
Method in Assembly A4 demands a permission P.
P
P is compared with grants of all callers on the stack above A4.

Understanding CAS

In a less secure but easier support model, principal-based security is often used and authorization is based on the identity of the user. This means that if an administrator launches an application it has full rights on the local machine. Unfortunately, if the administrator's identity is spoofed and a malicious user is able to execute code using the administrator's security context, the malicious user also has no restrictions. This is where code access security is important because it can provide additional restrictions and security based on the code itself, rather than the user running the code.

In the CDHS environment CAS is used to provide application isolation. This is very important because many of the Department’s web servers are shared or are in a shared security zone. Code on these servers is written by internal developers, contracted vendors and third-party commercial developers. Application isolation provides the following features:

- Isolate applications from each other. For example, code access security can be used to ensure that one Web application cannot write to another Web application's directories.
- Isolate applications from system resources. For example, code access security can restrict access to the file system, registry, event logs, and network resources, as well as other system resources.

.NET Built in Code Access Permissions Classes

The .NET Framework has many built-in code access permission classes that are designed to protect access to system resources. The built-in permission classes are listed in the following table.

Code access permission class	**Resource protected**
DirectoryServicesPermission	Directory services
DnsPermission	DNS services
EnvironmentPermission	Environment variables
EventLogPermission	Event logs
FileDialogPermission	File dialog boxes in the UI
FileIOPermission	Files and folders on the file system
IsolatedStorgeFilePermission	Isolated storage
MessageQueuePermission	Message queues
OleDbPermission	Databases accessed by the OLE DB data access provider
PerformanceCounterPermission	Performance counters
PrintingPermission	Printers
ReflectionPermission	Type information at run time
RegistryPermission	Registry
SecurityPermission	Execute code, assert permissions, call unmanaged code, skip verification, and other rights
ServiceControllerPermission	Running or stopping services
SocketPermission	Connections to other computers via sockets
SqlClientPermission	Databases accessed by the Microsoft SQL Server™ data access provider
UIPermission	Windows and other UI elements
WebPermission	Connections to other computers via HTTP

Although CAS provides a great deal of additional security, web applications are set to full trust by default. This means that all code has full rights by default. To restrict what the code can do, modifications to a .config file must be made. To modify code access security trust levels in ASP.NET, a switch in the machine.config or web.config is set and the application is configured as a partial-trust application. The following table lists the predefined trust levels for an ASP.NET application:

⇒ **Table VI ASP.NET CAS Trust Levels**

ASP.NET Trust Level	Main Restrictions
Full	Unrestricted permissions. Applications can access any resource that is subject to operating system security. All privileged operations are supported.
High	Not able to call unmanaged code Not able to call serviced components Not able to write to the event log Not able to access Microsoft Message Queuing queues Not able to access OLE DB data sources
Medium	In addition to the above, file access is restricted to the current application directory and registry access is not permitted.
Low	In addition to the above, the application is not able to connect to SQL Server and code cannot call CodeAccessPermission.Assert (no assertion security permission).
Minimal	Only the execute permission is available.

Each trust level is mapped to an individual XML policy file that controls the permissions granted to each trust level. The code access security policy is hierarchical and is administered at multiple levels. Policy can be created for the enterprise, machine, user, and application domain levels. ASP.NET code access security policy is an example of application domain-level policy.
Settings in a separate XML configuration file define the policy for each level. Enterprise, machine, and user policy can be configured using the Microsoft .NET Framework configuration tool, but ASP.NET policy files must be edited manually using an XML or text editor.

The individual ASP.NET trust-level policy files identify which permissions might be granted to applications configured for a particular trust level. The *actual* permissions that are granted to an ASP.NET application are determined by intersecting the permission grants from *all* policy levels, including enterprise, machine, user, and ASP.NET (application domain) level policy.
Because policy is evaluated from enterprise level down to ASP.NET application level, permissions can only be taken away. You cannot add permission at the ASP.NET level without a higher level first granting the permission. This approach ensures that the enterprise administrator always has the final say and that malicious code that runs in an application domain cannot request and be granted more permissions than an administrator configures.

The CDHS environment is configured in an n-tier model. The application server houses all business logic via web services. Because of this fact, CAS policies are configured on both the web server and the application server. This is accomplished by setting the security via the machine.config. The default standard for the ASP.NET policy permission is set to a trust level of ***medium***. Below is a table that lists the security settings allowed with a medium trust level set.

⇒ **Table VII CAS Medium Permission State Settings**

Permission and State	**Medium**
AspNetHosting Level	 Medium
DnsPermission Unrestricted	 3

EnvironmentPermission
Unrestricted
Read
Write
	TEMP; TMP;
	USERNAME; OS;
	COMPUTERNAME

EventLogPermission	
FileIOPermission Unrestricted Read Write Append PathDiscovery	 $AppDir$ $AppDir$ $AppDir$ $AppDir$
IsolatedStorageFilePermission Unrestricted AssemblyIsolationByUser- Unrestricted UserQuota	 3 3
OleDbClientPermission Unrestricted	
PrintingPermission Unrestricted DefaultPrinting	 3
ReflectionPermission Unrestricted ReflectionEmit	
RegistryPermission Unrestricted	
SecurityPermission Unrestricted Assertion Execution ControlThread ControlPrinicipal RemotingConfiguration	 3 3 3 3 3

Permission and State	**Medium**
SocketPermission Unrestricted	
SqlClientPermission Unrestricted	3
WebPermission Unrestricted	$OriginHost$

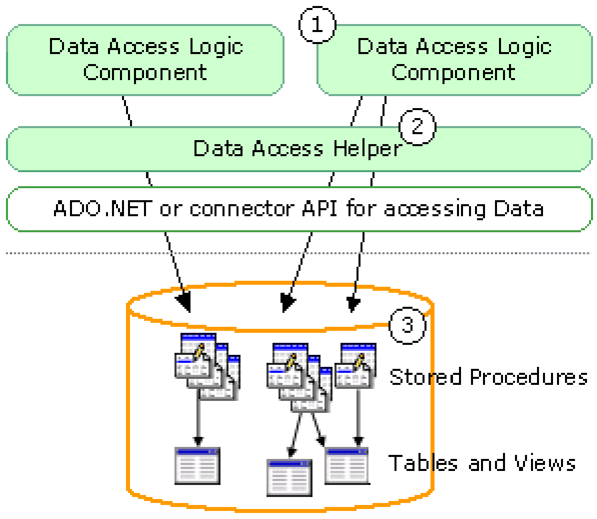
Setting a trust level of medium allows the applications and web services to run without having full access to privileged operations. This provides two major advantages: a reduced attack surface and application isolation. Since medium trust does not grant the application unrestricted access to all permissions, the attack surface is reduced by granting the application a subset of the full permission set. Many of the permissions granted by medium trust policy are also in a restricted state. If an attacker is somehow able to take control of the application, the attacker is limited in what he or she can do.

By restricting ASP.NET applications from having full access it makes compromising an application or running malicious code much more difficult. Configuring the machine.config makes it possible to enforce that all code running on the web server and application server maintain a minimum level of security. Again, this is important in a shared environment where code is being developed via a number of sources.

Data Access Layer

The Data Access Layer (DAL) forces application communication with the database to occur via stored procedures. The DAL provides methods for querying and updating the information stored in the database. The DAL creates an interface to the data allowing an application to have a simple programmatic way to retrieve and perform operations on the data. Each data access logic component typically provides methods to perform Create, Read, Update, and Delete (CRUD) operations relating to a specific business entity in the application (for example, order). These methods may be used by the business processes. Specific queries may be used by the user interface to render reference data (such as a list of valid counties). The following image illustrates a typical Data Access Layer (DAL).

⇒ **Figure 9 Typical Data Access Layer**


1
Data Access Logic Component
Data Access Logic Component
2
Data Access Helper
ADO.NET or connector API for accessing Data
3
Stored Procedures
Tables and Views

For example, if a user wants to update a record in the database a web service is called to complete the request. The DAL is passed the connection string, stored procedure name and any parameters that may be needed. The user's credentials are checked within the application layer. If the user belongs to a group that has appropriate rights, the DAL is invoked. If the user is not authorized then the request is denied.

SQL Server Injections

A SQL injection is a term describing the act of passing SQL code into an application that was not intended by the developer. They result from poor validation and coding practices that may result in the breach of a database. By using the data access layer and forcing use of stored procedures the threat of SQL injections is removed.

SQL injections are usually caused by using "string-building" techniques in order to execute SQL code. An example of this method of coding is listed below.

```
Set myRecordset = myConnection.execute("SELECT * FROM myTable WHERE
someText ='" & request.form("inputdata") & "'")
```

If someone wanted to cause some form of malicious code to run they could simple alter the code being sent to the database server. If the code or application was being executed by an account with a high security level (ex. SA, DBO) then the injection could potentially cause a great deal of harm. The following code demonstrates this.

Set myRecordset = myConnection.execute("SELECT * FROM myTable WHERE someText =' ' insert code that was not written by developer.)

By simply adding single quotes a hacker can cause the SQL Statement to change and bypass the original code. Next, the replacement code begins to execute.

Another problem resulting in the use of in-line SQL Statements is associated with database searches from an application. If the developer uses in-line SQL Statements to perform a search that uses a number rather than a string and does not perform a data input check, a SQL Statement can be injected. Often times the SQL statement that is injected gets executed. This causes an action to take place that was not intended by the developer.

Web Services vs. Remoting

Over time, it has become common practice to build applications as a set of components that are distributed across a network of machines and work together as part of one overall program. Browser-based Web applications, in contrast, are loosely coupled and remarkably interoperable. They communicate using HTTP to exchange MIME-typed data in a wide range of formats. Web services adapt the traditional web programming model for use from all sorts of applications, not just browser-based ones. They exchange SOAP messages using HTTP and other Internet protocols. Because web services rely on industry standards, including HTTP, XML, SOAP and WSDL, to expose application functionality on the internet, they are independent of programming language, platform and device.

The ASP.NET Web services infrastructure provides a simple API for web services based on mapping SOAP messages to method invocations. The clients of ASP.NET Web services do not have to know anything about the platform, object model, or programming language used to build them. The services themselves don't have to know anything about the clients that are sending them messages. The only requirement is that both parties agree on the format of the SOAP messages being produced and consumed, as defined by the Web service's contract definition expressed using WSDL and XML Schema (XSD).

.NET Remoting provides an infrastructure for distributed objects. It exposes the full-object semantics of .NET to remote processes using plumbing that is both very flexible and extensible. In order to use .NET Remoting, a client needs to be built using .NET.

The .NET Framework includes support for two distinct distributed programming models, web services and distributed objects.

The major differences between Remoting and XML Web services

⇒ **Table VIII Remoting vs. Web Services**

Remoting	Web Services
State full or stateless, lease-based object lifetime management	All method calls are stateless
No need for IIS (Although hosting in IIS/ASP.NET is recommended for security)	Must have IIS installed on the server
All managed types are supported	Limited data types are supported. For more information about the types supported by ASP.NET Web services, see the .NET Framework Developer's Guide on MSDN.
Objects can be passed by reference or by value	Objects cannot be passed
Contains an extensible architecture not limited to HTTP or TCP transports	Limited to XML over HTTP
Can plug custom processing sinks into the message processing pipeline	No ability to modify messages
SOAP implementation is limited and can only use RPC encoding	SOAP implementation can use RPC or document encoding and can fully interoperate with other Web service platforms. For more information, see the "Message Formatting and Encoding" section of the Distributed Application Communication article on MSDN.
Tightly coupled	Loosely coupled

Web Services

Web services are the fundamental building blocks in the move to distributed computing on the Internet. Open standards and the focus on communication and collaboration among people and applications have created an environment where XML Web services are becoming the platform for application integration. Applications are constructed using multiple XML Web services from various sources that work together regardless of where they reside or how they were implemented. Web services are software services exposed on the web through SOAP, described with a WSDL file and registered in UDDI.

The following table summarizes web services:

⇒ **Table IX Web Services Overview**

Web services build on the loose coupling of the traditional Web programming model, and extend it for use in other kinds of applications. There are three major differences between Web services and traditional Web applications: Web services use SOAP messages instead of MIME messages, Web services are not HTTP-specific, and Web services provide metadata describing the messages they produce and consume.
Microsoft's web services use standard web protocols such as HTTP, and data description languages such as XML, to exchange data over common ports, utilizing the HTTP infrastructure support that is already in place.
SOAP – Is a simple, XML-based protocol for exchanging structured and type information on the Web. The protocol contains no application or transport semantics, which makes it highly modular and extensible.
The SOAP Protocol format is standardized, so any application that understands SOAP can take advantage of Web Services.
Web Services Description Language (WSDL) - An XML-based contract language for describing network services offered by a server. This description includes details such as where we find the Web Service, what methods and properties that service supports, the data types, and the protocols used to communicate with the service. Tools can consume the WSDL, and build a proxy objects that clients use to communicate with the web services.

Web Services can be written in any of the .NET supported languages. The two most popular Microsoft languages are VB.NET and C#.

VOLUME V: APPLICATION ARCHITECTURE

This volume addresses the web based application architecture details. This includes the logical model, the physical implementation environment, coding practices and security requirements. It is critical to note that the logical model is applicable regardless of hosting location[1], however, these standards specifically address the implementation aspects of using Microsoft technologies in the CDHS environment or alternative hosting locations when using Microsoft technologies.

This volume is designed for the infrastructure and application development teams that will engage in deployment of applications for CDHS.

[1] It is important to note that the Information Security Policy should be read prior to hosting any information or application at an alternative location as there are requirements in doing this.

Section I Application Architecture Overview

CDHS has adopted industry standard architectures and continues to update the underlying infrastructure to stay current with changing technologies to avoid technology obsolescence. The core application architecture specifications are based on current technology, security requirements/considerations, management and performance. It is important to note that the Department has aligned its expertise and products with the Microsoft technologies, however, the logical design requirements must be adhered to regardless of product, hosting site and expertise.

New applications for CDHS must utilize this architecture in order to minimize deployment issues in the CHDS infrastructure. Outsourced application development efforts to be hosted at CDHS will be introduced in the lab, then the test and production environments and must use the same architectural model for design, development and deployment.

CDHS is aware of the resources and skill required to develop in the adopted model and accepts the business benefits of standardization to long-term viability of its applications. These benefits include, but are not limited to: increased data security, standard audit mechanisms, economy-of-scale by sharing resources and expertise, and logical and physical separation of business functions. Other benefits of a standardized development/infrastructure model include reduced costs through increased maintainability, increased opportunities to re-use components, and the increased ability to share data among programs. This separation of business functions includes the isolation of the presentation and business layer components simplifying upgrades and improving security.

The development and implementation of applications requires a lot of knowledge and expertise about the host environment. CDHS' ITSD can provide services to assist developers utilizing these standards to meet security and business requirements. For example, server build specifications, environment diagrams, desktop specifications, stored procedure generation, business layer component development, testing, design review, and hosting services are available. CDHS' ITSD often has instructional material and code samples for anyone that needs additional information developing systems based on this model.

CDHS has several web, application and database servers available for Programs to use that are already covered in ITSD operating costs, are maintained to support current industry standards and meet CDHS' security standards. These web servers exist in each of the segments available (Internet, Extranet & Intranet). There are also separate web servers available for development, test and production.

Developer Orientation

CDHS recognizes that developers inexperienced with the Departments infrastructure face technical challenges in getting started and fulfilling the core requirements. Based on our experience, first time CDHS developers have the highest risk of misinterpreting these standards, causing extra development work for the developer and infrastructure support staff.

CDHS offers a three-hour mandatory course explaining the CDHS hosting environment and standards. The objective is to assist developers in ensuring that applications are developed in accordance with these standards. This course not only helps first time developers understand the environment but also helps them understand why certain decisions have been made. This assists in ensuring successful project completion within estimated project timelines by eliminating conflicts with CDHS development standards.

This course will be offered monthly. To register, submit a request to the Internet Unit via your ITSD project manager

Third-Party Components

Third-party components are often used in development of projects for efficiency. The ITSD requires that these components be identified in the design phase and they must be reviewed and approved prior to procurement. The following additional requirements apply:

1. The components must be consistent with these standards.
2. The source code must be purchased with the components.
3. Maintenance agreements must be purchased when possible.
4. Funding must be available to upgrade or replacement obsolete or unsupported components.
5. Products must be purchased from companies demonstrating long-term viability.

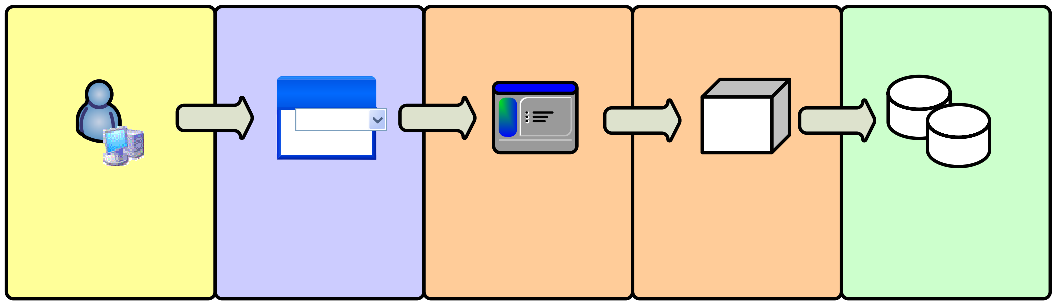
Commercial Solutions Selections

The delivery of full business solutions often requires the integration of multiple products. The selection of commercial products in the development of a complete business solution requires the review and approval of ITSD. The alternative products and architectures are to be identified in the Design phase and acceptance by ITSD is required before procurement is conducted.

Logical Model

The figure below, Logical Application Architecture, presents the logical view of the standard application architecture and is *applicable to any project* developed on behalf of CDHS. Subsequent sections define each of the logical layers and the role they perform.

⇒ **Figure 10 Logical Application Architecture**



By separating the code into layers CDHS creates the opportunity to modify the user interface separately from the rest of the code. This should give custom applications a longer "shelf life" within the Department.

End User

Most new applications developed for CDHS are being built using web-based technologies. The preference is based on the flexibility in providing access to distributed clients. The advantages of this thin client architecture with regard to client deployment, client resources, remote client access and standardized approach have been realized many times over. That does not mean, however, that all applications must be developed within this architecture.

The web client provides the mechanism for the customer to conduct communications with CDHS. CDHS has standardized on the Microsoft Windows client platform with Internet Explorer as the browser[2]. Any application that involves non-CDHS approved clients must be designed to support industry-leading browsers that include support of the .NET framework. Applications should not require client side components (ActiveX or Java Applets). If a client side component is necessary, prior authorization must be obtained.

[2] The development team should query the Client Technology Unit within the Information Technology Services Division to identify the lowest level browsers supported in the enterprise for employee accessed web applications. If a specific business unit will use the application, the local LAN Administrator should be contacted for this information For business partners, applications may use the lowest level browser supported by the customer. For public sites, State of California minimum requirements must be addressed.

Presentation Layer

The application model that CDHS has chosen to implement separates the presentation layer from the business layer components. The presentation layer typically contains graphics and the user interface.

For publicly accessible web sites, developers must adhere to the current CDHS implementation of the required State of California templates. Contact the Internet Unit for the current specifications.

Application/Business Logic Layer

The application layer components must be on a separate web server and exposed via web services. These application layer components should do the majority of work when it comes to implementing and enforcing the business processes that surround the application.

There are several different ways to implement the application layer technologies, but CDHS will only provide support for Web Services implemented on the Microsoft platform. Obtaining approval of your application design from the CDHS ITSD-IMAS Section through the reporting processes before coding will ensure that your application design will be supported through deployment

If you need to load support libraries for you web services components, please check with the Internet Unit to determine availability of existing components and compatibility of proposed components for the environment.

Data Access Layer

This layer is responsible for making the connection to the database server. This layer is also responsible for auditing user access to data (since all data requests for data must come through this layer). CDHS provides access to the data layer components using Microsoft's standard application blocks for web-based projects that access enterprise databases. The CDHS data access layer component has accompanying documentation and example applications using the Microsoft.ApplicationBlocks.Data.DLL

Data Layer

The data layer consists of Microsoft SQL Server products with the latest service packs and hot fixes. The data layer contains stored procedures configured to create, read, update and delete (CRUD) data.

The development and implementation of applications requires a lot of knowledge and expertise about the host environment. CDHS' ITSD can provide services to assist developers utilizing the standard models to meet security and business requirements. For example, server build specifications, environment diagrams, desktop specifications, stored procedure generation, business layer component development, testing, design review, and hosting services are available. CDHS' ITSD also has instructional material and code samples for anyone that needs additional information developing systems based on this model.

Section IV ITSD Hosted Application Infrastructure

CDHS' ITSD provides several services and resources to assist in developing and deploying an application meeting CDHS standards for applications. In addition to the infrastructure identified below, application design and development services are offered.

Distinct Security Zones to Meet Business Requirements

The network infrastructure is organized into three main security segments and the location of the user application is based upon who will access the application. The three segments are defined below:

Intranet - The internal CDHS network, accessible only by authorized CDHS staff. The Intranet zone is used by CDHS staff directly connected to the internal, private network. This includes the multiple locations that are directly connected to the CDHS LAN/WAN.

Extranet - An area of the network used primarily by non-CDHS staff, whose identity can be specifically identified and authorized for necessary levels of access. Access to the Extranet over public networks requires all communications to be encrypted at a minimum of 128-bits. CDHS business partners typically use this segment by connecting over the public Internet or the CDHS LAN/WAN and provide authentication information such as user names and passwords or certificates, to access their application. The Extranet segment is further subdivided into four isolated zones that provide additional separation between web, application, other interfaces and database servers.

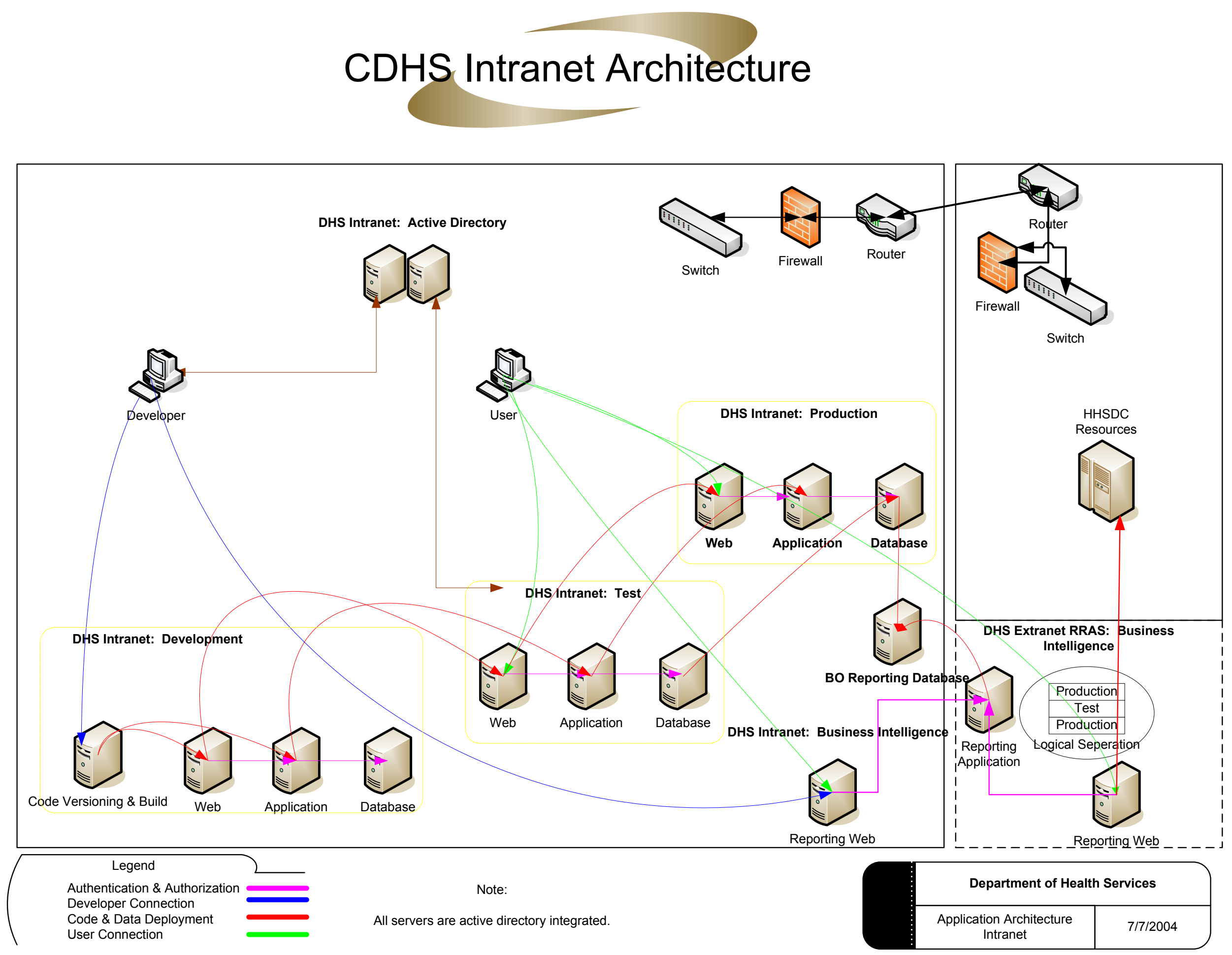
Internet - An area of the network accessible by anyone, whose identity may not be confirmed, and communications may or may not be encrypted. The Internet zone is typically used by the general public, connected over the public Internet. This zone will not contain or allow access to any data that is not supposed to be publicly available.

Please note that in some cases the application will need to utilize more than one segment of the infrastructure. This usually depends on the data access requirements, standards, and the type of users accessing the application business logic and data. For example, it is possible to have an business requirement to have CDHS personnel populating a database in the Intranet and have non-CDHS users viewing and reporting against a portion of this data. In this scenario, there is a need to develop and install application interfaces and components in both the Intranet and the Extranet or Internet zones. Since data must be accessible via multiple interface locations, application developers must design for this scenario. Infrastructure support staff must be available to assist in testing this scenario so that security is not compromised and administration is not increased.

Intranet

The Intranet is the term that describes CDHS' most secure private network. The users that are connected to the CDHS LAN/WAN who log into the CDHS domains may have access to applications and data on the Intranet servers. The client's CDHS account is used to authenticate users and authorize access to applications and data.

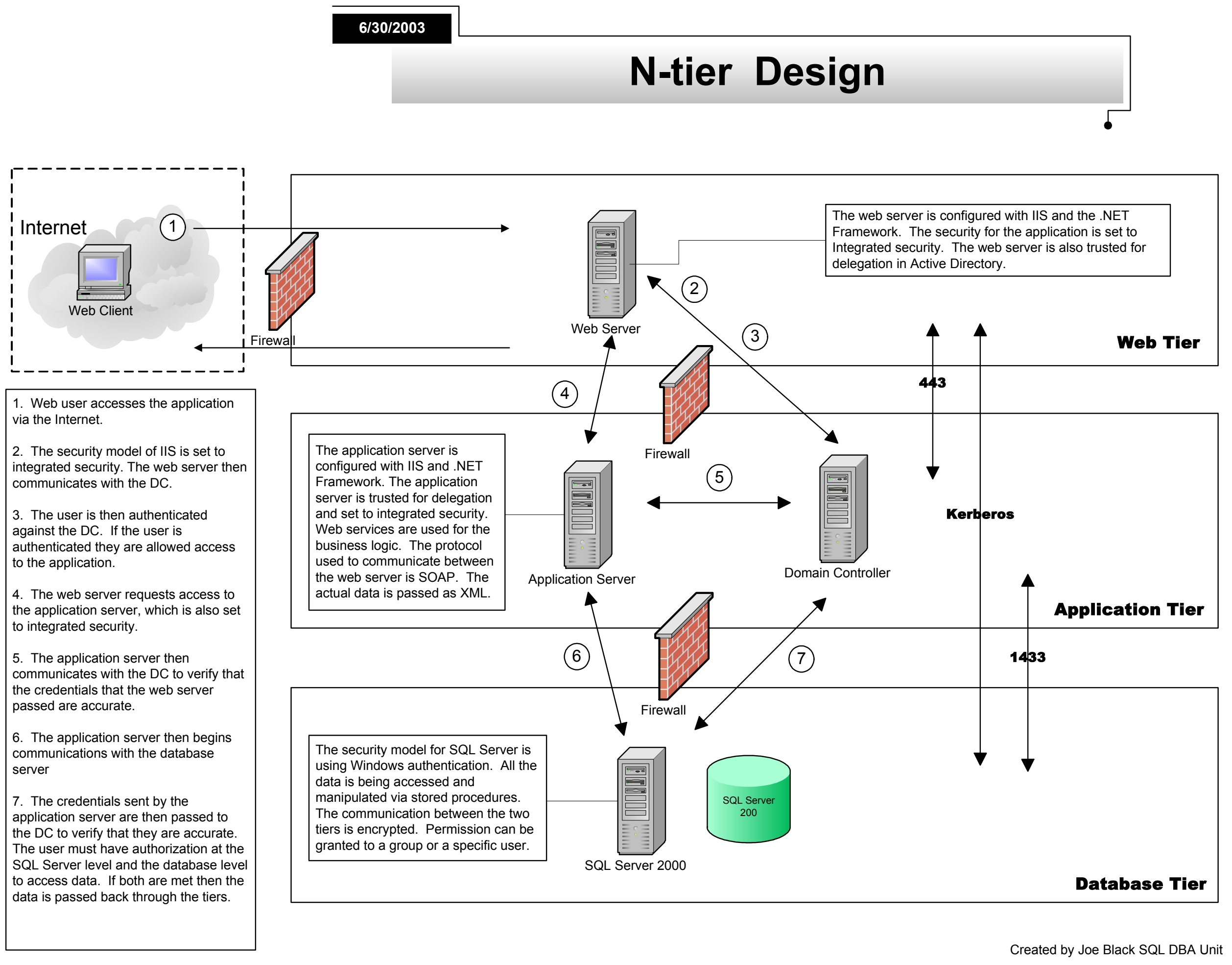
⇒ **Figure 11 CDHS Intranet Architecture**



Extranet

The Extranet segment is configured with separate web, application and database servers. Its purpose is to provide secure, authenticated and authorized access to CDHS applications and data for CDHS business partners.

⇒ **Figure 12 CDHS Extranet N-Tier Architecture**



The web server is configured with IIS and the .NET Framework. The security for the application is set to Integrated security. The web server is also trusted for delegation in Active Directory.

The application server is configured with IIS and .NET Framework. The application server is trusted for delegation and set to integrated security. Web services are used for the business logic. The protocol used to communicate between the web server is SOAP. The actual data is passed as XML.

The security model for SQL Server is using Windows authentication. All the data is being accessed and manipulated via stored procedures. The communication between the two tiers is encrypted. Permission can be granted to a group or a specific user.

1. Web user accesses the application via the Internet.

2. The security model of IIS is set to integrated security. The web server then communicates with the DC.

3. The user is then authenticated against the DC. If the user is authenticated they are allowed access to the application.

4. The web server requests access to the application server, which is also set to integrated security.

5. The application server then communicates with the DC to verify that the credentials that the web server passed are accurate.

6. The application server then begins communications with the database server

7. The credentials sent by the application server are then passed to the DC to verify that they are accurate. The user must have authorization at the SQL Server level and the database level to access data. If both are met then the data is passed back through the tiers.

As shown in the above figure, the CDHS Extranet consists of several zones which are logically and physically protected from each other. For example, the business logic layer hosts the data access layer which is isolated from the database server via the firewall.

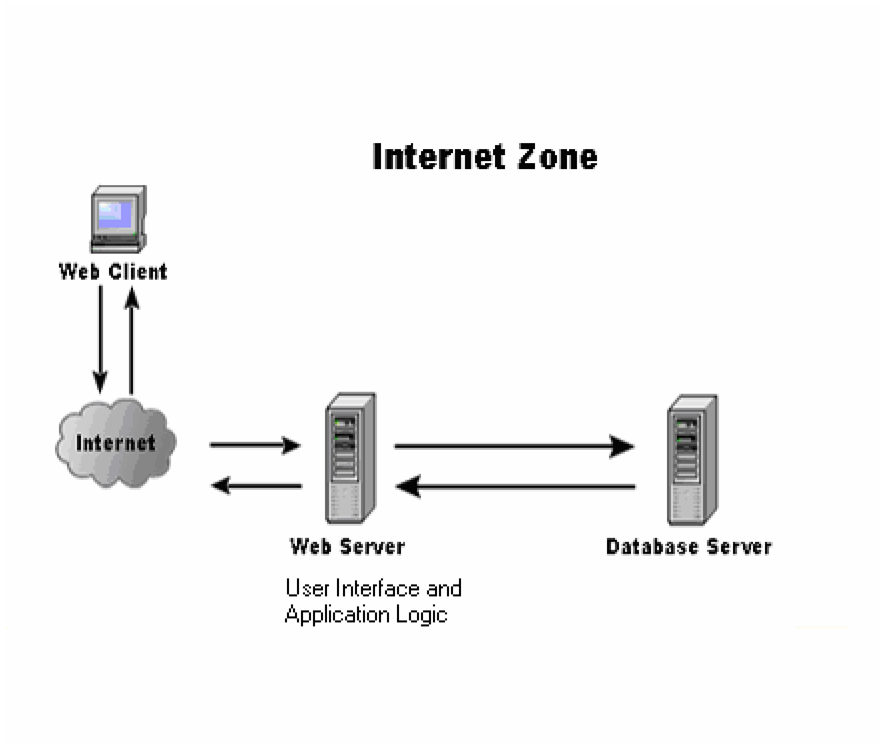
The Extranet has its own Active Directory Forest and Domain. Developers writing applications for the Extranet must account for transaction and reporting interfaces for CDHS Intranet users in the application design. Active Directory organizational units are created and delegated to Program to manage. Group permissions must be applied to web and application servers by the IU. Account and group administration is then delegated to Program or the CDHS IT Help Desk staff to the extent possible using web based utilities for administration.

Internet

The purpose of the Internet segment is to support public access to non-confidential information until further business requirements are identified. This segment protects internal CDHS resources at the network and application layers.

The Internet zone provides test and production web, application and database servers. The servers on this segment do not have the ability to directly communicate with database servers or domain controllers on the Intranet segment. The only connectivity occurs via push of data and content from the Intranet segment to the Internet segment.

⇒ **Figure 13 CDHS Internet Architecture**



The database servers in the Internet segment have a conduit for data replication with a replication server on the Intranet segment. The database replication server resides on the Intranet and provides CDHS employees reporting and data manipulation capability for data on the Internet segment. Not all data within a database must be replicated. Horizontal and vertical data partitioning is available and provides additional security.

Application and presentation updates occur via designated replication channels. See the section on the Code Deployment for instructions on code promotion.

In addition, the internal CDHS domain does not have a trust with the public domain. Therefore, Internet applications cannot use CDHS internal accounts based on this security model. User accounts are created within SQL server, within the internet domain, or through lightweight directory access protocol (LDAP) compliant methods. This security model protects the private CDHS network from the users on the public Internet.

Internet SQL Servers cannot be directly accessed from the Internet. All access to the data on Internet zone SQL Servers are made via test or production web application servers. The Internet SQL Servers are not the database of record.

Distinct Servers for Application Life Cycle Management

The ITSD provides the support for the following types of servers to support an application throughout its life cycle.

Development

Development systems have been configured and made available so that the application programmers that are working on a project can have a central repository to store their work. It is expected that most developers will use their desktop systems to begin development; however, this is not required.

Development systems for each of the zones will only be made available on the Intranet segment. Developers will be provided with access to web, application, database and version resources. Developers will not be provided with administrator level permissions on any of the servers in the shared environment as the resources are shared between multiple development teams. Operator level permissions will be granted when necessary. There are several projects being developed and each environment must be kept autonomous. Therefore, one development team will not be given the ability to affect the work of another development team.

The Development environment use is based on rules so that the integrity of the environment can be preserved for all utilizing the available service. Contact the SQL Server Unit and Internet Unit units for the most current rules. Examples of some of the rules include:

1. Developers have database owner (DBO) access to databases.
2. All objects created must be owned by DBO.
3. SSU Groups will be created with a single account for developer testing
4. Full backups will be configured daily on databases
5. The database recovery model will be simple (No transaction log backups)

6. Over 5 min. period, no database can utilize over 20% of the resources of the system
7. Only developers will be allowed access to the development systems. User testing will be done in the test environment.
8. Publishing roles will be established for web application posting.
9. Component registration will be scheduled in advance.
10. Standard maintenance intervals will be conducted and developers will be subject to scheduled downtime.

Test

Test systems are available for use and will be based on test plans. Test servers are available for each segment. Project test users will have access to the same infrastructure connectivity and security model that the project will experience in production. Movement from development to test will be orchestrated between web and database teams. This involves the creation of security accounts, configuring web and application servers and transferring data to the test systems.

The test environment use is based on rules so that the integrity of the environment is preserved for all utilizing the available service. Contact the SSU and IU units for the most current rules. Examples of some of the rules include:

1. Rebuild SYSDEPENDS from development when testing begins
2. Developers will have DBO access to the database during troubleshooting
3. Groups will be created with users for testing
4. Full database backups will be configured daily
5. The recovery model will be simple (No transaction log)
6. Test plan must be submitted in advance, agreeing upon timeline
7. Database will be taken off-line after conclusion of test plan
8. Performance monitoring of the systems during testing will be conducted
9. Over 5 min. period, no database can utilize over 20% of the resources of the system
10. Test users must be identified in advance
11. Component registration will be coordinated in advance

The CDHS' ITSD technical teams may also help with the testing of an application to determine the overall impact of the application on the environment. CDHS owns tools for this purpose. The results of this testing will be shared and recommendations will be made based upon this imperial data.

Production

Production systems are configured and maintained to provide the highest level of availability for CDHS' customers. Once testing is complete, development and infrastructure teams resolve any issues regarding maintenance, data import/export functions, scheduled processes, etc. The ITSD technical teams are responsible for backups, operating system and application level maintenance functions for the infrastructure of the application. Support services by Program for testing infrastructure upgrades must remain available through the system life cycle.

The development group will not have database owner (DBO) permissions to the production databases. Access to the data within a database will be granted through the web services components and the application only. All database schema changes must be coordinated through the SSU Group to ensure integrity of the database and application. Change control processes will be adhered to, ensuring the integrity of the application.

The Production environment use is based on rules so that the integrity of the environment can be preserved for all utilizing the available service. Contact the SSU and IU units for the most current rules. Examples of some of the rules include:

1. Only the SSU Group has DBO Access to the database
2. All groups will be created and implemented
3. Connection accounts' password will be reset
4. Full database backups configured daily (special backup requirements can be requested of the SSU Group)
5. Database Recovery model is Full (for Trans log backup)
6. Full transaction log backup configured an hour interval
7. Change control procedures must be followed for production changes
8. Over 5 min. period, no database can utilize over 20% of the resources of the system

CDHS Research Center (CDHSRC)

The CDHSRC provides an opportunity for the organization to evaluate the business benefit of identified technologies prior to procuring them and to evaluate custom and commercial products prior to implementing them in the core development, test and production environments.

The key objectives of the CDHSRC are to:

1. Establish an environment for IT issues related to current technology and future advancements in research and computing.
2. Assist decentralized IT support staff in analyzing IT needs and act as a catalyst for bringing leading edge technology to LAN Administrators.
3. Establish functionality within the ITSD structure to promote an educational environment for CDHS Programs .
4. Provide a development environment for the purposes of planning and testing to meet the IT needs of the organization.

Scheduling Time and Resources

The lab manager can best accommodate lab scheduling when given two weeks or more advanced notice of an upcoming project. Complicated projects requiring multiple pieces of equipment or network design may require a month prior notice. When contacting the lab manager, provide the following:

1. List of customer owned equipment to be brought into the lab for project.
2. List of lab equipment required for the project.
3. Architectural requirements for project (i.e. special access to resources outside the lab etc.)
4. Desired start and end date.
5. List of approved users to access test equipment.
6. Special requirements not provided by the laboratory.

Maximum Project Duration

Projects may last up to two (2) months in the CDHSRC. However, if the participant has brought in their own equipment, other than the use of shared infrastructure equipment, the project may go as long as four (4) months.

Requesting Extra Time

Lab customers may request an extension to continue work in the lab, however, there is no guarantee of an extension. In the event that extra time is needed, contact the lab manager. The lab manager will review requests and attempt to assist where possible.

Commercial-off-the-Shelf Product Hosting

CDHS' environment also includes provisions for hosting products purchased to meet specific business requirements. Among these products available in the shared web hosting environment are:

1. Remedy
2. Business Objects
3. WebTrends
4. WatchFire (Accessibility and Quality)
5. SAS

The IU, SSU and ISO will work with Programs in selecting, testing and deploying web-enabled products for the enterprise. The NTSS, SSU, and ISO will work with Programs in selecting, testing and non-web based solutions for the enterprise.

Enterprise-Reporting

The implementation of Business Objects**,** an enterprise-reporting tool provides CDHS Programs with the ability to access and disseminate their data via the web to internal CDHS staff and selected external customers.

The Department supports a wide range of legacy reporting system technologies and fields many proposals for new reporting systems. Most proposals require their own servers and infrastructure, a Feasibility Study Report (FSR), and are often very expensive to maintain and support. The availability of enterprise reporting tools that access information in different types of database products led CDHS to find a more efficient method for reporting and data analysis.

The enterprise reporting tool selection is the result of a departmental collaborative effort to identify a solution for meeting an increased demand for access to and dissemination of CDHS information. The effort required that the solution meet the needs of a variety of customers, internal knowledge workers and external business partners.

The products key features utilized in a number of production systems include:

1. It's ability to access different database structures without added programming.
2. Built-in capability to deliver information over the Extranet and Intranet.
3. Wide variety of report generation levels of detail for batch and ad hoc reports.
4. Minimum training required for programs to customize reports for their customers.
5. Ability to use a common reporting tool for a variety of CDHS programs.
6. Ability to access information stored at HHSDC as well as CDHS.
7. Ability to be supported by the ITSD DBA Unit for use throughout CDHS.

Section V Server Standards for Hosting Applications

The CDHS supports a multi-tier server build specification. The base build for all servers, Tier I, is developed by the Server group. The Internet and SQL server groups support installation and configuration of web, application and database components, actively maintaining a Tier II build. The minimum server build standards for web, application and database servers must obtained from the IU and SSU respectively. These standards do change, so it is best to check with the IU and SSU for the most current builds.

Specific Server Configuration Requirements

The following tables list the settings required in the n-tier configuration for web, application and database servers. This configuration is required for servers in the Intranet and Extranet.

Web Server

⇒ **Table X Web Server Configuration Requirements**

Configure IIS	
Step	**More Information**
Disable Anonymous access for your Web application's virtual root directory Enable Basic Authentication for your Web application's virtual root	
Configure ASP.NET	
Step	**More Information**
Configure your ASP.NET Web application to use Windows authentication	Edit Web.config in your Web application's virtual directory Set the <**authentication**> element to: <authentication mode="Windows" />
Configure your ASP.NET Web application for impersonation	Edit Web.config in your Web application's virtual directory Set the <**identity**> element to: <identity impersonate="true" />

Application Server (that hosts the Web service)

⇒ **Table XI Web Application Server Configuration Requirements**

Configure IIS	
Step	**More Information**
Disable Anonymous access for your Web service's virtual root directory Enable Windows Integrated Authentication for your Web service's virtual root directory	
Configure ASP.NET	
Step	**More Information**
Configure your ASP.NET Web service to use Windows authentication	Edit Web.config in your Web service's virtual directory Set the <**authentication**> element to: <authentication mode="Windows" />
Make sure impersonation is off	Impersonation is off by default; however, double check to ensure that it's turned off in Web.config, as follows: <identity impersonate="false" /> Note that because impersonation is disabled by default, the same effect can be achieved by removing the <**identity**> element.

SQL Database Server

⇒ **Table XII SQL Server Configuration Requirements**

Step	**More Information**
Create a SQL Server account for each application.	
Configure SQL Server for Mixed mode	
Establish database permissions for the database user	Grant execute rights on the stored procedure

Active Directory Configurations

The following procedures are implemented in Active Directory to support the n-tier model in CDHS:

⇒ **Table XIII CDHS Active Directory Configuration Requirements**

Step	More Information
Servers Trusted for Delegation	Server must be configured in Active Directory as "Trusted for Delegation"
Add User accounts to Active Directory	All users of the application must have an account in Active Directory and it must also be configured for delegation.
Create Groups for Role Based Security	Management groups must be set up for each application. These groups will match the applications roles used for securing the application.

Port Configuration Requirements

The following minimum specifications for port configurations must be in place:

⇒ **Table XIV CDHS Minimum Firewall Configuration Requirements**

Port	Description
80/TCP -- HTTP	Hypertext Transfer Protocol is the set of rules for exchanging files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web. Relative to the TCP/IP suite of protocols (which are the basis for information exchange on the Internet), HTTP is an application protocol.
88/UDP -- KERBEROS	Kerberos is a network authentication system which is based on the key distribution model. It allows entities communicating over networks to prove their identity to each other while preventing eavesdropping or replay attacks. The Kerberos Key Distribution Center (KDC) listens on this port for

Port	**Description**
	ticket requests. Port 88 for Kerberos can also be TCP/UDP
389/TCP -- LDAP	LDAP is the Lightweight Directory Access Protocol. LDAP is designed to be a standard way of providing access to directory services. LDAP is the primary way the operating system accesses the Active Directory database.
443/TCP -- HTTPS	HTTPS is A variant of HTTP used for handling secure transactions. HTTPS is a unique protocol that is simply SSL underneath HTTP.
636/TCP – LDAP OVER SSL	LDAP over Secure Sockets Layer (SSL). When SSL is enabled, LDAP data that is transmitted and received is encrypted.
1433/TCP - SQL[3]	Used to communicate with SQL Server

[3] Check with the SQL Server Unit for the correct port specifications.

Section VI Application Security Model Requirements

CDHS has chosen an application security model that closely follows Microsoft's current recommendations for an n-tier application that uses web services in the application zone. The intention is to create an environment that is as secure as possible minimizing restrictions for developers as much as possible.

Authentication

There must be an ability to verify that the user attempting to access your application is who they say they are. The supported methods in the CDHS environment are described below.

Authorization

Intranet (runs exclusively within the CDHS internal domain) authorization occurs when each user of the application has a network account enabling the user's to gain access to authorized resources based on the credentials passed and validated by the domain controller. Logged on users will not need to log into the application once validated on the network.

Extranet (access for trusted business partners via the Internet) authorization occurs when a user attempts to access CDHS resources is prompted to provide credentials that must be validated against the account information, and subsequent resource access permission, stored within a domain controller in the extranet zone.

Internet (access for the general public) authorization is not currently required by any CDHS web-enabled business function and therefore anonymous authorization is permitted to gain access to resources.

Trusted Sub-System Model

CHDS adopts the Trusted Subsystem Model so that the original identity of the user is checked at the IIS/ASP.Net gate, mapped to a role, and then authorized based on role membership of the user. System resources for the application are then authorized at the application or role level.

The multi-account setup is used for CDHS applications. The idea behind this model is that users are added to groups based on their work tasks. For example, a group of users may need to simply view data. A reports group can be created. This group only needs read access to the data. When this group accesses the data within SQL Server they are doing so through a single Windows account. This

Windows account only has read access to the data within SQL Server. When developing an application using this model it is very important to clearly identify what groups need to be set up and what permissions they need.

Enabling Authentication and Authorization in CDHS Applications

Intranet – An Intranet application is designed to run exclusively within the CDHS internal domain. In this scenario, each user of the application has a CDHS network account which they have used to log into the network that has been validated by the domain controller. Based on this scenario, the following is done:

First, configure the web server to use Windows Integrated Security in IIS thus the credentials of the user logged into the computer are used. The credentials are not to be stored within the application or database.

Next, the ASP.NET application is configured to use the default authentication, Windows. This authentication information is held in the web.config configuration file. This XML document can be easily edited with notepad or any other editing tools. The web.config file reflects the following:

```
<configuration>
    <system.web>
        <authentication mode="Windows" />
    </system.web>
</configuration
```

Internet – Internet applications are accessed via the Department's public web servers. For information that must be available to the public that requires no authorized access, there are no accounts set up on the network for authentication purposes. This means that we grant anonymous access to these sites. There are cases when we need to secure some of the information on the web site. To accomplish this, we authenticate the user before allowing them to have access to secure areas of the site.

Anonymous

First, IIS is configured for Anonymous access. Next, the ASP.NET application is configured to use no authentication. The web.config file reflects the following:

```
<configuration>
    <system.web>
        <authentication mode="None" />
    </system.web>
</configuration
```

Authentication, Forms-Based

Forms-based authentication is used for providing authenticated access to a public web application. The user provides their credentials through a custom login page. The credentials are then automatically checked for verification. If the user is authenticated a cookie is created. Each additional secure page then uses this cookie for access. If the cookie exists, the page can be accessed by the requesting party. If the cookie does not exist, the requestor is automatically rerouted to the login page.

First, IIS is configured to allow Anonymous access to the site. Anonymous access is acceptable because the authentication is handled by the application using forms-based authentication.

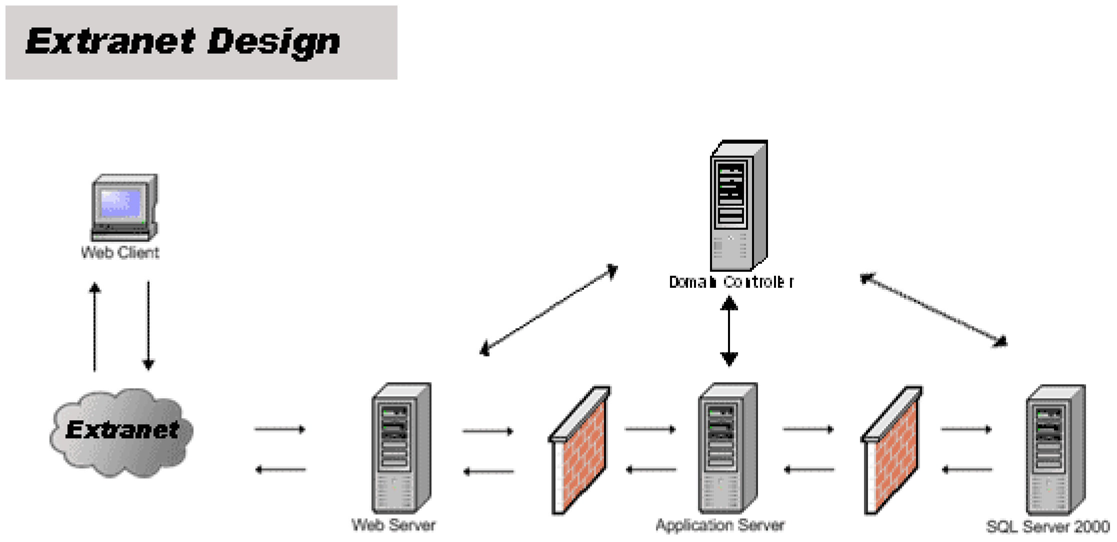
Next, the ASP.NET application is configured to use forms authentication. This authentication information is held in the web.config configuration file. This XML document can be easily edited with notepad or any other editing tools. The web.config file reflects the following:

```
<configuration>
        <system.web>
              <authentication mode="Forms" />
       </system.web>
</configuration
```

Storage of user credentials can be accomplished in multiple ways including the web.config (for a small number of users), an XML data source (small to medium number of users) or a database (large volume of users). CDHS will only support storage of user credentials in a database for Internet applications.

Extranet – An Extranet application is one in which trusted business partners will be using the application. The application and resources are exposed over the Internet, which means that a secure method of authentication is required. Generally, this means tighter control over user accounts compared to the Internet and a higher level of trust with the user. The following illustration shows a typical layout for a CDHS Extranet application.

⇒ **Figure 14 CDHS High-Level Extranet Design**



CDHS's Extranet will host new ASP.NET applications. There is a domain controller (DC) with Active Directory configured. All Extranet application users must have an account in the DC. The user supplied credentials are passed to the DC. If the supplied credentials are verified the user is permitted to access the site. Secure Socket Layer (SSL) is to be used therefore ensuring that all communication is encrypted. The following steps are applied:

First, IIS is configured with Basic authentication and Anonymous access disabled. This prevents public access to the site and prompts the user to login. The user then enters their supplied credentials for authorization.

Next, the ASP.NET application is configured to use the default authentication, Windows. This authentication information is held in the web.config configuration file. This XML document can be easily edited with notepad or any other editing tools. The web.config file reflects the following:

```
<configuration>
     <system.web>
          <authentication mode="Windows" />
     </system.web>
</configuration
```

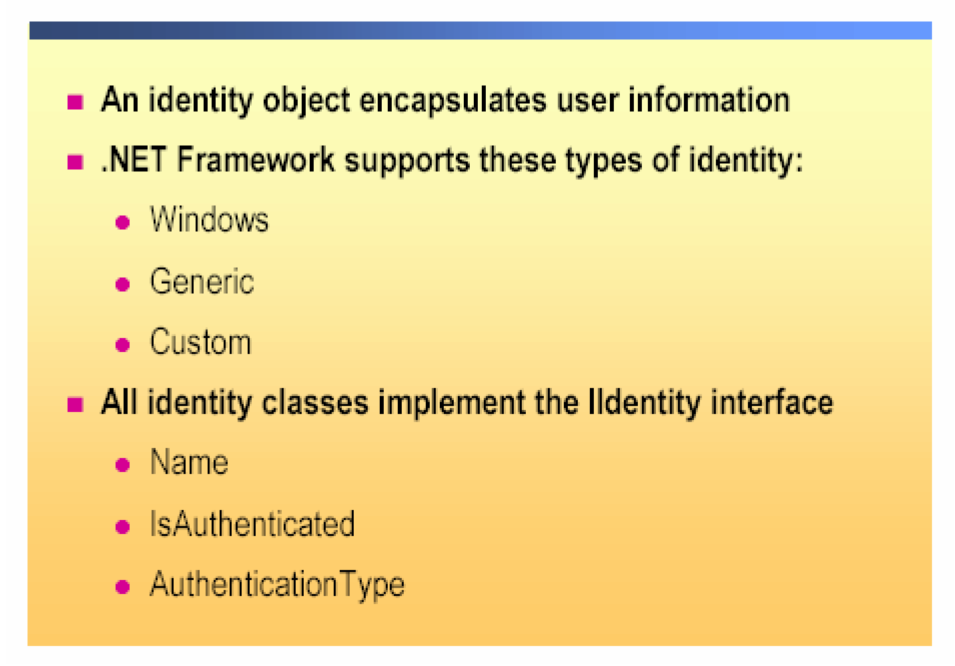
Understanding and Enabling Role-Based Security

Role-based security in the .NET Framework extensively uses two concepts: identities and principals.

Identity & Principal

An *identity* encapsulates the user's logon name. A *principal* encapsulates the membership information of the user's role. The runtime provides functionality to perform authorization checks by using identity and principal-related objects directly, or by using imperative or declarative permission checks.

⇒ **Figure 15 Identity**



- An identity object encapsulates user information
- .NET Framework supports these types of identity:
  - Windows
  - Generic
  - Custom
- All identity classes implement the IIdentity interface
  - Name
  - IsAuthenticated
  - AuthenticationType

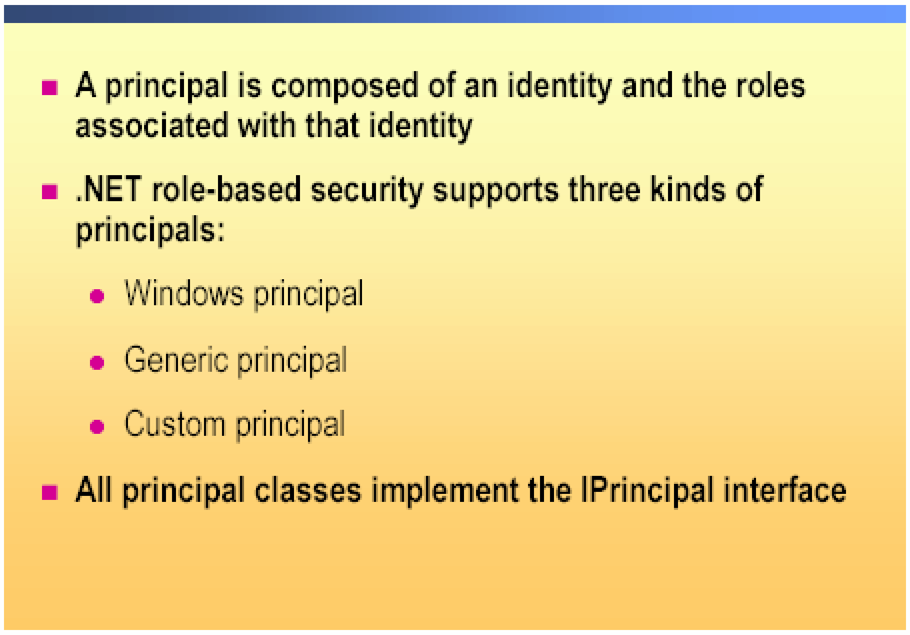
A *principal object* represents the security context under which code runs. This includes the identity of the user, as represented by an associated identity object, and the roles associated with the user.

Role-based security in the .NET Framework supports three kinds of principals, as described in the following table.

⇒ **Table XV CDHS High-Level Extranet Design**

Principal	Description
Windows	Represents Windows users and their roles. The roles are the Windows groups of which the user is a member. The WindowsPrincipal class implements this principal.
Generic	Represents users and roles that exist independent of Windows users and their roles. Essentially, the generic principal is a simple solution for application authentication and authorization. The GenericPrincipal class implements this principal.
Custom	Represents application-specific role information. Any custom principal class must implement the IPrincipal interface.

⇒ **Figure 16 Principal**



- A principal is composed of an identity and the roles associated with that identity
- .NET role-based security supports three kinds of principals:
  - Windows principal
  - Generic principal
  - Custom principal
- All principal classes implement the IPrincipal interface

All principal classes implement the IPrincipal interface. The IPrincipal interface has an Identity property that stores the identity object that is related to the current principal and an IsInRole method that determines whether the current principal belongs to the specified role.

Windows Identity

To perform role-based validation you must first create a WindowsIdentity object and a WindowsPrincipal object. The code you write to handle these objects will vary depending on whether code must validate only once or repeatedly.

⇒ **Figure 17 Role Based Validation Code Sample**

```
' Visual Basic
' Get the current Windows identity
Dim myIdentity As WindowsIdentity = _
   WindowsIdentity.GetCurrent()
```

```
' Visual Basic
' This code continues from the previous sample
' Create a windows principal from the current identity
Dim myPrincipal As New WindowsPrincipal(myIdentity)
```

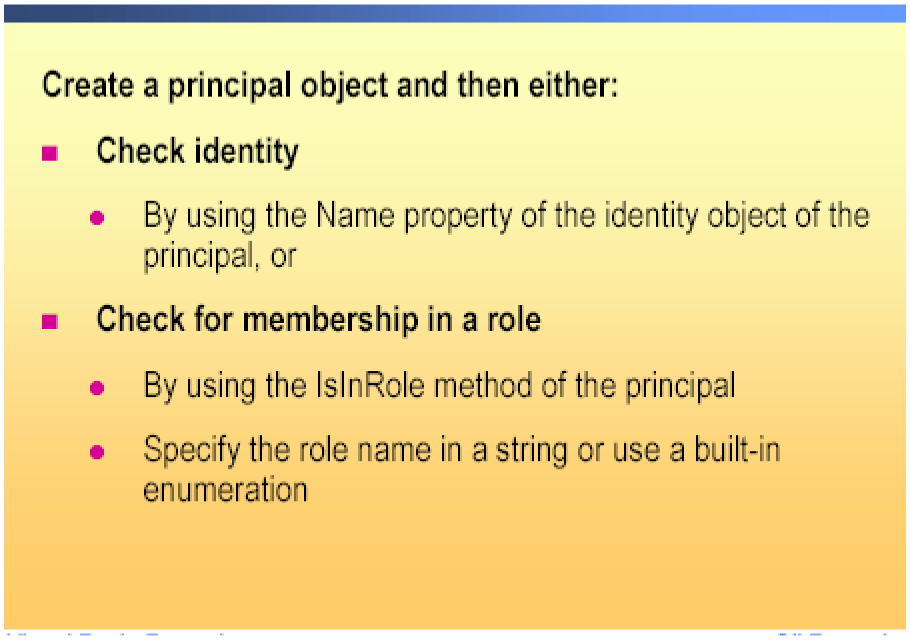
```
' Visual Basic
' Set the principal policy
AppDomain.CurrentDomain.SetPrincipalPolicy( _
   PrincipalPolicy.WindowsPrincipal)
```

```
' Visual Basic
' This code continues from the previous sample
' Get the current windows principal
Dim myPrincipal As WindowsPrincipal = CType( _
   System.Threading.Thread.CurrentPrincipal, _
   WindowsPrincipal)
```

Checking Identity and Role Membership

You can use principals and identities to control access to code based on the user's identity and role membership.

⇒ **Figure 18 Checking Identity & Role Membership**



Create a principal object and then either:

- Check identity
  - By using the Name property of the identity object of the principal, or
- Check for membership in a role
  - By using the IsInRole method of the principal
  - Specify the role name in a string or use a built-in enumeration

The following example uses a case-insensitive string comparison to see if the name of the user is DOMAIN\Fred. The first two parameters to the static String.Compare method are the strings to be compared, and the third parameter tells the method to do a case-insensitive comparison.

⇒ **Figure 19 Case Sensitive String Compare Code Sample**

```
' Visual Basic
' Assume a valid Principal is in myPrincipal
If [String].Compare(myPrincipal.Identity.Name, _
   "DOMAIN\Fred", True) = 0 Then
   ' Permit access to some code
End If
```

You can check role membership by calling the IsInRole method on the principal object. For example, you can use this technique to determine whether the user belongs to the DOMAIN\Administrators role.

For WindowsPrincipal objects, a role corresponds to a Windows group, including the domain. When checking for membership in built-in Windows groups, you can use the WindowsBuiltInRole enumeration. Using this enumeration allows you to develop code that is more easily localizable.

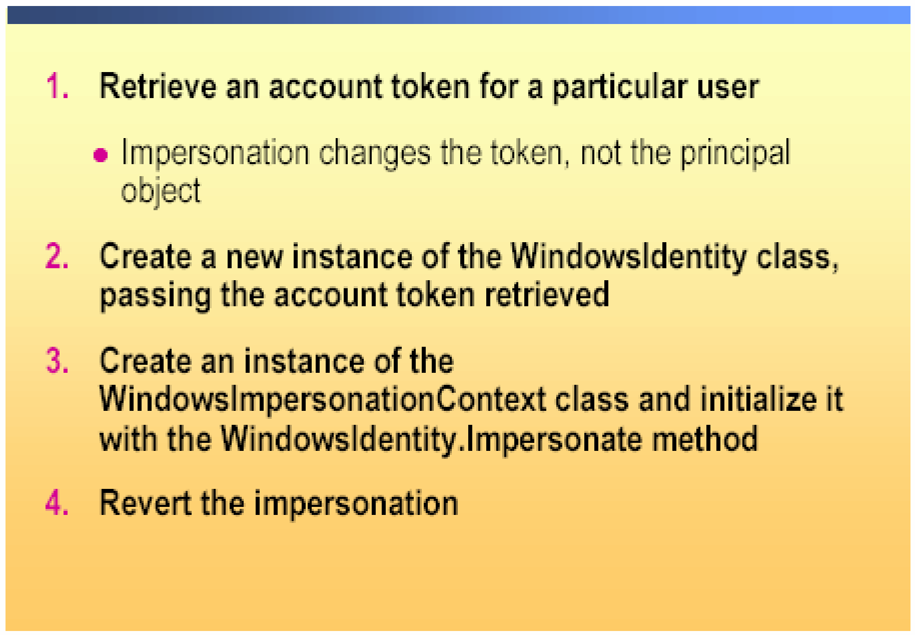
⇒ **Figure 20 WindowsBuiltInRole Enumeration Code Sample**

```
' Visual Basic
' Assume a valid Principal is in myPrincipal
If myPrincipal.IsInRole("BUILTIN\Administrators") Then
   ' Permit access to some code
End If
```

Impersonation

To achieve impersonation, the following steps must be applied.

⇒ **Figure 21 Impersonation**



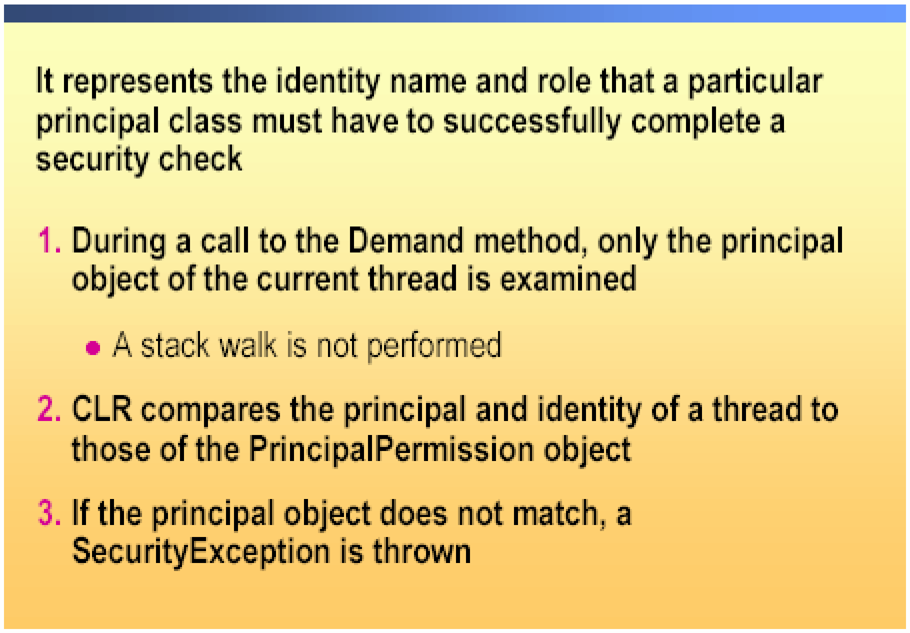
1. Retrieve an account token for a particular user
   - Impersonation changes the token, not the principal object
2. Create a new instance of the WindowsIdentity class, passing the account token retrieved
3. Create an instance of the WindowsImpersonationContext class and initialize it with the WindowsIdentity.Impersonate method
4. Revert the impersonation

Principal Permission Object

You can make role-based security checks by creating and demanding PrincipalPermission objects that represent identity names and roles for which you want to check.

The PrincipalPermission object represents the identity name and role that a particular principal class must have to complete a security check.

⇒ **Figure 22 Principal Permission Object**



It represents the identity name and role that a particular principal class must have to successfully complete a security check

1. During a call to the Demand method, only the principal object of the current thread is examined
   - A stack walk is not performed
2. CLR compares the principal and identity of a thread to those of the PrincipalPermission object
3. If the principal object does not match, a SecurityException is thrown

During a call to the Demand method, the common language runtime examines the current thread's principal and identity objects to determine whether the identity and role information match those represented by the PrincipalPermission object on which Demand is being called. If the principal object does not match, a SecurityException is thrown. A stack walk is not performed, because only the principal object of the current thread is examined.

⇒ **Figure 23 Creating and Demanding Principal Permission Code Sample**

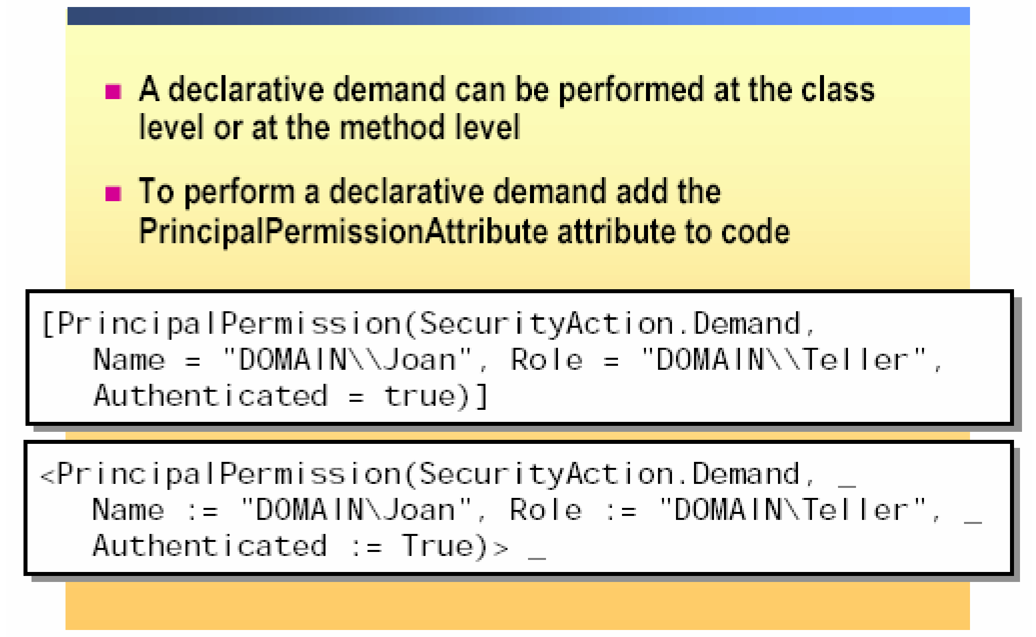
```
' Visual Basic
' Create the principal permission
Dim principalPerm As New PrincipalPermission( _
   "DOMAIN\Joan", "DOMAIN\Teller", True)
```

```
' Visual Basic
' Demand the principal permission
Try
   principalPerm.Demand()
   ' if this code is reached, the demand succeeded
Catch
   ' if this code is reached, the demand failed
End Try
```

Declarative Role-Based Security Checks

You can use the PrincipalPermission attribute to declaratively demand that users running your code belong to a specified role or have a specific identity name.

⇒ **Figure 24 Declarative Demand**



- A declarative demand can be performed at the class level or at the method level
- To perform a declarative demand add the PrincipalPermissionAttribute attribute to code

```
[PrincipalPermission(SecurityAction.Demand,
   Name = "DOMAIN\\Joan", Role = "DOMAIN\\Teller",
   Authenticated = true)]
```

```
<PrincipalPermission(SecurityAction.Demand, _
   Name := "DOMAIN\Joan", Role := "DOMAIN\Teller", _
   Authenticated := True)> _
```

You can place demands at the class level as well as on individual methods. If you place a declarative demand at both the class and member levels, the declarative demand at the member level overrides the demand at the class level.

⇒ **Figure 25 Principal Permission Demand Code Sample**

```
' Visual Basic
<PrincipalPermission(SecurityAction.Demand, _
   Name := "DOMAIN\Joan", Role := "DOMAIN\Teller", _
   Authenticated := True)> _
```

If you omit the Name, Role, or Authenticated property from the attribute instantiation, the check will match any identity name, role, or authentication status, respectively. If the current thread does not contain a matching principal a SecurityException is thrown.

Implementing Role-Based Security for CDHS Applications

CDHS' method for implementing role-based security for its applications utilizing web services requires authorizing callers. Each of the following methods may apply at the method level and are listed in the order of preference:

- Declarative Demand
- IsInRole() method
- Imperative Demand

If possible, use declarative syntax so a systems administrator will be able to easily determine the permissions that the application requires. For example, use this at the method level:

```
[PrincipalPermissionAttribute(SecurityAction.Demand,  Role = "User")]
```

In addition, allow and deny roles on a per application basis in web.config such as:

```
<authorization>
  <allow roles="Admin, Tester, Auditor" />
  <deny users="*” />
</authorization>
```

Passing Credentials for Authentication to Web Services

When calling a web service, the developer does so by using a web service proxy; a local object that exposes the same set of methods as the target web service. The developer can generate a web service proxy by using the wsdl.exe command line utility. Alternatively, the proxy can be generated by adding a web reference to a Visual Studio.NET project.

Note: If the web service for which the developer wants to generate a proxy is configured to require client certificates, the developer must temporarily switch off the requirement for client certificates while adding the reference, otherwise an error occurs. Having added the reference, the developer must remember to reconfigure the service to require certificates. An alternate approach is to keep an offline WSDL file available to consumer applications. The developer must remember to update this if the web service interface changes.

Specifying Client Credentials for Windows Authentication

If the security zone dictates the use of a Windows authentication scheme, the developer must specify the credentials to be used for authentication via the **Credentials** property of the web service proxy. If the developer does not explicitly set this property, the web service is called without credentials resulting in an HTTP status 401, access denied response.

Using Default Credentials

Client credentials do not flow implicitly. The web service consumer must set the credentials and authentication details on the proxy. To flow the security context of the client's Windows security context (either from an impersonating thread token or process token) to a web service the developer sets the **Credentials** property of the Web service proxy to **CredentialCache**.**DefaultCredentials** as shown below.

```
proxy.Credentials =
System.Net.CredentialCache.DefaultCredentials;
```

Adopting this approach allows for the following:

- This flows the client credentials when using NTLM, Kerberos, or Negotiate authentication.
- If a client-side application (for example a Windows Forms application) calls the web service, the credentials are obtained from the user's interactive logon session.
- Server-side applications, such as ASP.NET web applications use the process identity, unless impersonation is configured in which case the impersonated caller's identity is used.

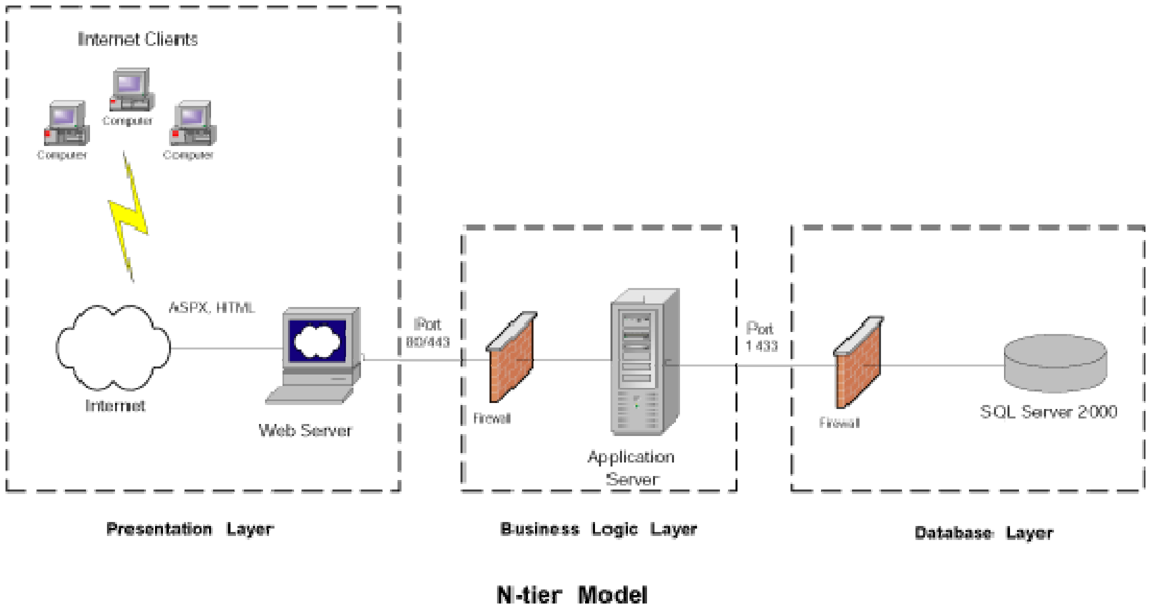
Set the PreAuthenticate Property

The proxy's **PreAuthenticate** property can be set to true or false. Set it to true to supply specific authentication credentials to cause a **WWW-authenticate** HTTP header to be passed with the initial web request. This saves the web server from denying access of the initial request and performing authentication on the subsequent request.

Business Logic Layer (BLL)

The following paragraphs describe the physical architecture layout of the BLL and the model for developing code in this tier. This is designed for developers that have experience programming in an n-tier environment, and are also familiar with .NET and Web services. The goal is to clearly detail the physical architecture and the coding model for this layer within the CDHS environment.

⇒ **Figure 26 N-Tier Model**



The business logic layer will contain business rules that govern interaction with data. In the business logic layer there is no need to house any knowledge of the database or data store. All of the data interaction occurs in the data layer. In the BLL there is a separation between the layers. This allows the changes to be made to different layers without affecting the other layers

In the CDHS environment the BLL contains all of the business rules and error handling for the application. The BLL does not have any knowledge of the data layer, which will allow for changes to be made without affecting any of the other layers. Web services are used to pass the data to the presentation layer using the SOAP protocol over http.

After a web service is created and is deployed to the BLL server the presentation can consume the web service. Once the web service is consumed the class is exposed and calls can be made from the presentation layer. All of the functionality of the web service can then be used throughout the application by simply making a call to the desired namespace.

As seen from the n-tier illustration above, specific ports will be open between the presentation and business logic layers. With the removal of port 1433 this in essence eliminates direct communication with the database. The removal of direct communication is a major security benefit.

As ports 80/443 are the only ports open for communication between the presentation and BLL, the data passed to the presentation layer will use the corresponding protocol. CHDS supports web services in the BLL to handle all of the application logic, rules and error handling. CDHS supports VB.NET as a preferred language.

Once the business logic is created the web service must be able to communicate with the presentation layer. For communication to occur between the presentation and BLL the web service must be located and the communication channel must be established.

Data Access Layer

The DAL is the logic used to access data stored in SQL Server. All CDHS applications that access data stored in SQL Server will do so using stored procedures and no underlying tables are to be exposed, greatly reducing the possibility of data being accessed outside of an application.

The following methods further eliminate the possibility of SQL injections and are achieved through the use of the Data Access Layer (DAL):

- In an n-tier model the application layer no longer has the ability to communicate directly with the database eliminating the ability to use in-line SQL Statements. The Data Access Layer (DAL) forces applications to communicate with the DAL rather then with the database via stored procedures. A single account is created for the DAL to use in making connections to the database. The database is configured to use this fixed identity. This account has access to the database and has the appropriate rights to interact with the database. Creating a fixed identity accomplishes the following database security benefits:
- Users do not have logon rights to the database.
- The single account can be stored in a secure location.
- The single account can be configured with a strong password.
- All interactions to the database are via stored procedures.
- Users can not connect directly to SQL Server with any tools such as Access, Excel, etc.
- Stored procedures are used to access and interact with the database. If all interactions with the database occur through stored procedures no access to the base tables can occur.
- When using stored procedures, implement them using the ADO command object so that variables are strongly typed.
- No in-line SQL statements are to be used.
- Normal users are created (no fixed server or database roles) which provides the ability to natively access all objects in the database to which

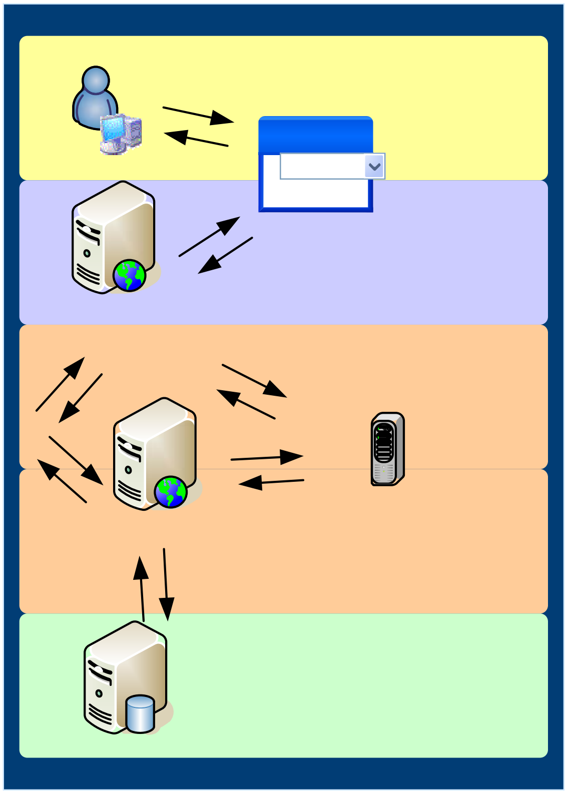
the account is given access. At best, this may mean only being able to run some stored procedures. At worst, this means possible read/write access to all tables and views.

- Utilize an extensive code review and testing process that exposes any potential problems.

Applying in the Intranet Zone

In the Department's current Intranet design there are four main application layers. The main layers are Presentation, Application, Data Access, and Data. The Intranet Zone is set up to use Active Directory, and it is designed as a single forest with no trust to any other zone. The following image is a high level diagram of the Department's Intranet design.

⇒ **Figure 27 CDHS Application Layers in the Intranet**



The web applications rely on SQL Server as a primary data store. When data is accessed, inserted or updated certain steps need to be taken. To complete these steps an understanding of the configuration of the Intranet zone is important. In the Intranet zone must have a user account in the Active Directory. This enables us to use integrated security rather than store the password some place in the application or in a database table.

When a user accesses an application their credentials are verified and if they are authenticated they can access the application. If the user accesses a portion of the site that needs to interact with SQL Server, a web service in the application layer is called. The user's credentials are then passed on to the application layer. When the web service makes a call to the data layer it is doing so on behalf of the user.

When the DAL is called, the user's credentials are passed from the application layer. When the DAL makes a call to SQL Server in the data layer it is doing so using the credentials of the user. All of the interactions with SQL Server are through stored procedures. Stored procedures can be secured by only giving execute rights to certain individuals or groups, so the user must either have execute rights or belong to a group that has execute rights to interact with SQL Server.

For example, if the user wants to update a record in the database the web service is called to complete this request. The DAL is passed the connection string, stored procedure name and any parameters that may be needed. The user's credentials are then checked and if they are authorized to execute the stored procedure then the process takes place. If the user is not authorized then the request is denied.

The design of the DAL is created in such a way that data components are exposed to the application layer. When the web service needs to interact with the Data layer they do so by calling the desired component. For example, if the application needs a dataset to be returned to populate a datagrid in the presentation layer the data component designed to return a dataset is called. This allows the detailed steps of ADO.NET to be handled at the DAL and removes the need for the application developer to deal with these complexities. The following code sample demonstrates a web service requesting a dataset:

```
' DataSet that will hold the returned results
        Dim ds As DataSet

        ' Call ExecuteDataset static method of SqlHelper class that returns a Dataset
        ' We pass in database connection string, command type, stored procedure
name and a "1" for CategoryID SqlParameter value
        ds = SqlHelper.ExecuteDataset(ConnectionString, _
CommandType.StoredProcedure, "getProductsByCategory", _
New  SqlParameter("@CategoryID", 1))

        ' Get XML representation of the dataset and display results in text box
        txtResults.Text = ""
        txtResults.Text = ds.GetXml()

        DataGrid1.Visible = True
        DataGrid1.DataSource = ds
        DataGrid1.DataBind()


    End Sub
```

The code shows how the SqlHelper component is called and returns a dataset. The DAL hides all of the code to open a connection, execute a command, pass a parameter and close the connection. An application developer that is coding the application layer does not need to worry about all of the underling tasks. They just need to simply call the DAL component.

Applying in the Extranet Zone

The Extranet design is similar to the Intranet. In the Extranet there is central domain controller that is running Active Directory. Each user permitted to access applications in the Extranet Zone must have an account in Active Directory. The Extranet Zone is also configured as a forest with no trust relationships. However, a key difference between the Intranet and Extranet is the way in which users are initially authenticated. Because Extranet users will be accessing the application from the Internet there is no way of knowing who they are. In an Intranet environment the users log into their computer on the network before accessing the application, therefore we know who they are logged on as. To get the identity of the Extranet user we must have them provide us with some form of identification. This process is accomplished by setting up basic authentication in IIS. When the user accesses an application set up with basic authentication they are prompted with a login screen that they use to provide their credentials. These credentials are then validated against Active Directory, and if the user's credentials are good they have been authenticated and can access the application.

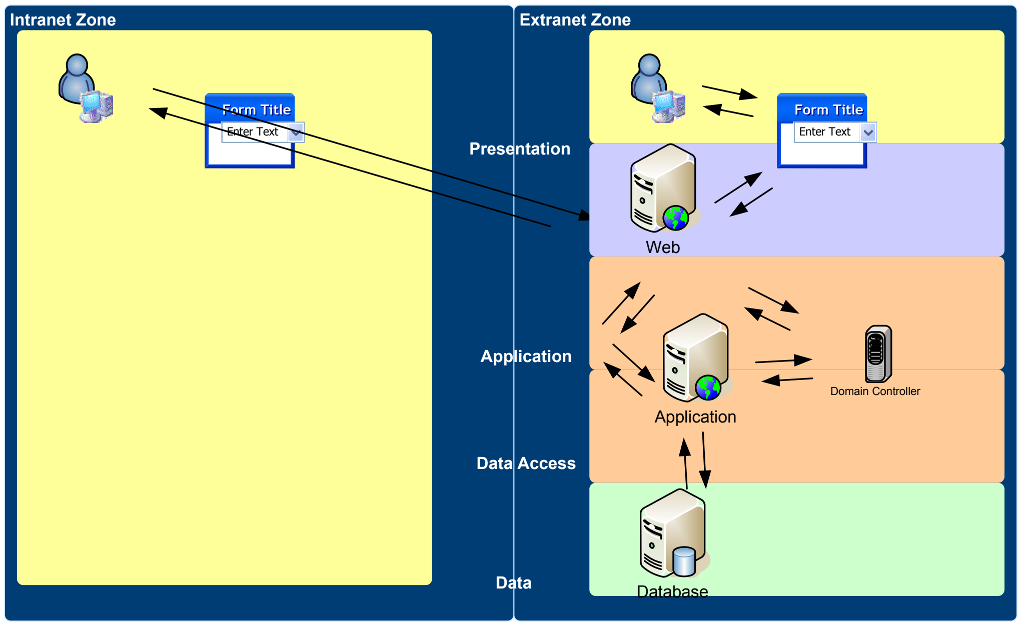
The basic steps of how the application communicates through the multiple layers are the same as in the Intranet Zone.

Intranet/Extranet Applications

There are cases when an application needs to be accessed from both the Intranet and Extranet. In these cases the models mentioned above must be modified. The reason that the applications that are accessed from both zones need some modification is because of the lack of trust between the two zones. In the models mentioned above the credentials of the users can be passed from layer to layer. The reason that this is possible is because of the trust relationship between layers.

In the scenario when an Intranet user needs to access to an Extranet application there is no trust between the Intranet and Extranet zones. This means that if the Intranet credentials were to be passed into another zone they would be rejected. The following image represents both zones and the communication between them.

⇒ **Figure 28 Intranet / Extranet Application Interfaces**



In the image above the database only resides in the Extranet An Intranet user will be given an Extranet account to access the application. Intranet accounts do not have permissions to access Extranet data. When the DAL attempts to communicate with the data layer the original credentials could be used sacrificing the ability to use connection pooling. To allow for connection pooling, a standard SQL Server account is created. When the DAL communicates with SQL Server the connection string that is passed will contain the login information of the standard account.

When developing an application in either the Intranet or Extranet Zone it is important to understand how to access the data through the DAL. Often times an application can have interfaces that reside in both the Intranet and Extranet zones. In this case the developer must understand the security and credential requirements that are in place. Because there is no trust between the two zones the user's credentials can not be passed through all layers.

Code Access Security

Code Access Security (CAS) is the method of allowing the administrator to control the level of permission granted to a given assembly based on the origin of the identity and origin of the assembly. In each of the three CDHS zones specific CAS policies are set to further secure the applications being hosted on a ***shared*** server. It is very important in a ***shared environment*** to control what the code from each application has access to.

CDHS CAS Policy

CDHS has chosen to implement CAS within the machine.config file. The following specifies the configuration applicable in the CDHS environment.

Web Servers (Medium Trust)

1. All web applications will run in "Medium" mode.
2. All web applications will be limited to calling pre-defined web services only.
3. Any code that needs more permission to execute will be contained in a web service.
4. Under unique circumstances the Internet Unit may grant the developer the right to create a *Custom Configuration* file for use by the web application.

Default Configuration

- Machine.Config
  - set Trust Level = "Medium"
  - set AllowOverride = "false"
- Web.Config
  - Set urlOrigin= "Url to web service(s) to be called" to limit the services the web app may call into.
  - Impersonate = True

Application Servers (Full Trust)

1. All web services will run in medium mode by default as set in the web.config file.
2. If a service requires more permission to execute there are two options:
   a. If the service requires full trust then make no entry in web.config, using the default setting of full in the machine.config.
   b. If the service does not require full trust but requires more than the default permissions, the developer will create a custom configuration files incorporating the minimum level of permissions required to execute. Once reviewed and approved by the Internet Unit, the web.config will be set to use the custom file.

Default Configuration

3. Machine.Config
    - Set Trust Level = “Full”
    - Set AllowOverride =”true”
  - Web.Config
    - Set Trust Level = “Medium”
    - Impersonate = false
    - If Special Permissions are required
      - Set Trust Level= “Custom Config File”
      - OR for applications needing full trust leave this entry blank

CDHS Standards for .NET Applications

The following sections are intended to provide developers with the requirements for developing .NET Applications in the CDHS environment. This includes topics such as naming conventions, creation of web services and database references. Applications developed by CDHS staff or by vendors on behalf of CDHS programs are expected to follow these standards.

Coding standards, clearly commented code and structured code all contribute to help others to understand and comprehend the program more easily. It is also easier to debug and maintain a clearly written application.

Coding styles may differ between programmers; however, the following rules should be applicable to most coding styles. It is also important to note that CDHS has adopted these standards from Microsoft.

General Overview

Coding standards, clearly commented code and structured code all contribute to help others to understand and comprehend the program more easily. It is also easier to debug and maintain a clearly written application.

Coding styles may differ between programmers and this document should not be considered to enforce any particular style. All the following rules should be able to be applied to all styles of coding.

CDHS Coding Standards

The following address the coding standards applicable in the CDHS environment.

Exception Handling & Custom Error Pages

Exception Handling

1. Bubble up all unhandled exceptions to the UI layer

   Display a generic fixed message to users

   Write Exception information to either the event log or email it

2. Catch only specific exceptions that you can handle

    - Re-Throw any exceptions you cannot handle to the UI layer
    - Do Not catch the general Exception class error except for in the UI layer
3. Provide a central Error Handler for your application (in a base class)
4. Catch exceptions in both Global.asax (Application_Error) and in your Web Page (base class)
5. Do not overuse Try Catch blocks as they slow down processing. If you catch exceptions in your page events then there is no reason to catch any exception that you cannot handle in any helper classes. Library classes that are used by many other applications should always use this rule and let the caller decide what to do about exceptions.

Custom Error Pages

Do not allow exception details to propagate from your Web applications back to the client. A malicious user could use system-level diagnostic information to learn about your application and probe for weaknesses to exploit in future attacks.

The **<customErrors>** element can be used to configure custom, generic error messages that should be returned to the client in the event of an application exception condition. The error page should include a suitably generic error message, optionally with additional support details. You can also use this element to return different error pages depending on the exception condition.

Make sure that the mode attribute is set to "**On**" and that you have specified a default redirect page as shown below:

<customErrors mode="On" defaultRedirect="YourErrorPage.htm" />

The **defaultRedirect** attribute allows you to use a custom error page for your application, which for example might include support contact details.
Note Do not use **mode="Off"** because it causes detailed error pages that contain system-level information to be returned to the client.

If you want separate error pages for different types of error, use one or more **<error>** elements as shown below. In this example, "404 (not found)" errors are redirected to one page, "500 (internal system errors)" are directed to another page, and all other errors are directed to the page specified on the **defaultRedirect** attribute.

```
<customErrors mode="On" defaultRedirect="YourErrorPage.htm">

   <error statusCode="404" redirect="YourNotFoundPage.htm"/>

   <error statusCode="500" redirect="YourInternalErrorPage.htm"/>
```

```
</customErrors>
```

For an example of using a central error handling see:
DemoServiceSolutionVB.sln

Input Validation

- Validate all user input for script tags, field length and data type on the *Server*
- HTML encode any data input by users that is to be displayed in the UI
- Test for empty strings using System.String.Length using:

```
            If (Name.Length == 0 )    NOT:  If (Name == “” )
```

Asserts and Demands

Asserts require security demands. Asserting security permission without performing any security checks can leave an exploitable security weakness in code.

```
try {
            SomePermission.Demand();
            SomePermission.Assert();
            Work();

    } finally {
            SecurityPermission.RevertAssert();
    }
```

Assemblies Specify Permission Requests

Add attributes specifying what permissions your assembly will demand, might demand, and what permissions it does not want granted. For example, the following attribute indicates that an assembly will, at minimum, require read access to the USERNAME environment variable:

```
[assembly:EnvironmentPermissionAttribute(SecurityAction.RequestMini
mum, Read="USERNAME")].
```

To specify permissions that the assembly might demand, use

```
SecurityAction.RequestOptional.
```

To specify permissions that the assembly must not be granted, use

SecurityAction.RequestRefuse.

Transfer to Page

DO NOT use **Server.Transfer** (which bypasses authorization) to go to a secure page, use **Response.Redirect** instead

Comments

Comments make the code more comprehensible.
Comments should explain the algorithm or logic being used.
Simple comments should NOT be used as it just clutters up the code, the variable names should help to explain any simple code.

Example 1 (c#):

```
// get the user name
string   name            = getUserName();
// get count of sick days used
int      days            = getSickDaysUsedt( name );
```

The above comments do not serve any useful purpose

Example 2 (vb):

```
‘  may be null if the user is not yet logged in
string   name            = getUserName();

‘  may be negative is user has used all sick days plus some
int      days            = getSickDaysUsedt( name );
```

This example at least offers some insight not gleaned from the function names themselves

For more complicated scenarios, use more comments in blocks as:

(c#)

```
/* ---------------------------------
   This code ASSUMES:
         1. The user is logged on
         2. The user has admin rights
         3. The userName is never NULL
--------------------------------- */
```

```
bool  canWork          =  canUserWork( userName );

// ---------------------------------
//   Create a sorted list as the rules require all names to be in sorted
//   order by name.
//   List Format expected by callers is:
//          Key=Name   Value=SS#
// ---------------------------------
SortedList lst  =  getUserList();
```

Function Header Comments

All functions should have a header explaining the general operation, parameters and return values. Also include any special assumptions or requirements.

This is a full standard header:

```
/* -----------------------------------------------------------
Function:        UpdateUser

Description:     To update the user in the AD using LDAP

Parameters:      SortedList lstAdd     - groups to add user
to
                 string     adsPath    - full AD path of
user

Return Value:   string Id  - the Id of this user

Notes:           Throws exception if caller is not an admin
                 Updates all fields except username

Assumptions:     adsPath is never null

Side Effects:    removes the user from the global update
                 list when done


Modifications:
By                           Date            Purpose
-----------------            --------        ----------------
-----john doe                     12/12/03        Initial
Creation

-----------------------------------------------------------
- */
```

Simple get/set functions can use a simple header rather than the full header above

```
/* ---------------------------------------------------------
Function:        getUserList

Description:     returns a list of users in alpha order
                 List may be empty but never null
---------------------------------------------------------
*/
private SortedList getUserList()
     { return (SortedLIst) Session(KEY_LIST) }
```

Creating Web Services and Database References

It is very important to create all references in a way that they can be easily modified as the code is compiled and moved through the development, test and production process. In the current ITSD n-tier hosting environment the presentation layer will access all business logic though web services, so it is very important that the web proxies are created to use dynamic URLs. To allow for the web proxies to communicate correctly with the proper web services the configuration files must point to the proper location. This is accomplished through the use of an itsd.config file. This file will be managed by the IU for use in deploying solutions. This will allow the IU to deploy successive solutions in a more robust and consistent way.

The following steps detail the process of creating and calling web services via web references that are controlled via an itsd.config file.

1. Set the **URL Behavior** property of your Web service references to **dynamic** to gain maximum configuration flexibility both within the development and production environments.
2. In the web.config file set file="ITSD.CONFIG in the appSettings node as: <appSettings file="itsd.config">
3. Create a sub folder in your project named ITSD and place an ITSD.CONFIG file in it using the below format
   a. Copy any items in your web.config file that need to be modified when deployed into the itsd.config file. (IU will modify your entries correctly when deployed.)

For Web Services this will usually contain items such as a connection strings

```
<?xml version="1.0" encoding="utf-8" ?>

<appSettings>

<add key="YOUR_CONNECTION"  value="Data Source= Catalog=… />
```

```
</appSettings>


For Web Apps this will usually contain items such as a URL(s) to web services

<?xml version="1.0" encoding="utf-8" ?>

<appSettings>

<addkey="Demo.DemoService" value="http://localhost/Demo/DemoService.asmx" />

</appSettings>
```

4. Check this file into SourceSafe.

Update this file if you add or remove any items from your web.config file that also need to be in this file. Notify the Internet Unit if this file has been modified to add new keys or delete old keys.

Connecting to the database in the current ITSD n-tier model is conducted via the Data Access Layer (DAL). ITSD has chosen to handle all connections to SQL Server via the DAL, which is a modified version of Microsoft's Microsoft.ApplicationBlocks.Data.dll. The connection settings will be created in the same manner as listed above for web services

CDHS Best Practices for .NET Applications

The following sections are intended to provide developers with general best practices for developing .NET Applications regardless of hosting entity. Applications developed on behalf of or by CDHS staff are expected to follow these practices whenever feasible and not in conflict with CDHS Standards.

This information is available on Microsoft's web site by searching for "**Patterns and Practices**" and may not reflect the adopted methods for hosting applications at CDHS. See References for a list of documents that provide a more detailed view of the items presented here.

Microsoft Application Blocks

Overview

An application block is a *fully tested* .NET component written by *experienced Microsoft .NET programmers*. Each comes with *full source code* and is available for *free* on the Microsoft web site. Using Application Blocks reduces the amount of custom code needed on a project and avoids duplicating many hours of programming, testing and debugging, that Microsoft has already done.
See http://msdn.microsoft.com/library/en-us/dnbda/html/daab-rm.asp

Data Access Application Block (for SQL Server only)

The Data Access Application Block contains optimized data access code that will help you call stored procedures and issue SQL commands against a SQL Server database.

Exception Management Application Block

The Exception Management Application Block provides an extensible framework for handling exceptions. You can easily log exceptions to the Event Log, other data sources or notify operators, without affecting your application code.

Database Access

Data Access Block

If using SQL Server than always use the MS - Data Access Block described above.

Connection Pooling

Always take advantage of connection pooling by using the same exact *connection string* on all connections made in an application.

1. Setup a new user account on the DBMS to be shared by all connections (always use the lowest privileges needed)
2. Store the connection string encrypted in the web.config file, so it can be shared and also be secure
3. OPEN Connections just before using and always CLOSE connections as soon as you are done. (best to put code in a **Using** block to be sure connections are closed)

Data Reader

If you are just reading data and do not require disconnected recordsets, do not need to update the data and do not need to move randomly through the data, then always use the Data Reader class for speed (provides a forward only *Fire Hose* cursor).

Preventing SQL Injection Attacks

1. Always use the parameters collection, do not concat sql statements from user input
2. Replace apostrophe with double apostrophe: input.Replace("'", "''")
3. Limit the length of any user input to the max length of the column being queried
4. Remove any sql comment characters "—" from the input
5. Always run using a least privileged account. (Use the minimum privileges needed only)

Definitions

Pascal Casing Defined

This convention capitalizes the first letter of each word, as in BackColor.

Camel Casing Defined

This convention starts each word in lowercase and then capitalizes the first letter of each word, as in backColor.

Project Namespaces

Use a common root namespace name. The root namespace into which you place your types (structures, classes, interfaces, and so on) should match the project and assembly name. This should also match the folder structure for the project. While .NET does not require this alignment, it makes sense to synchronize names because it then becomes easy to tell which types live in which assemblies.

Note: Microsoft Visual Basic® .NET projects expose the root namespace via project properties. By default, any type created within the Visual Basic project will be placed inside this namespace. **If you use explicit namespace statements in your Visual Basic .NET project, delete the root namespace entry**, otherwise the explicit namespace name is appended to the root namespace name.

C# projects expose a default namespace property via project properties. This is again used to determine the namespace into which new types added to the project are placed. However, unlike Visual Basic .NET projects, the root namespace is explicitly stated via namespace statements within your source files.

Example of Namespace matching Assembly name:

```
    Namespace: MyCompany.Utilities.Data
    Assembly:   MyCompany.Utilities.Data.dll
```

Example of Namespace matching directory structure:

```
    IUProjects.ProjectName
    IUProjects.ProjectName.Data
    IUProjects.ProjectName.Data.SqlServer

    The projects directory structure on disk should match this layout:
    D:\Inetpub\Applications\IUProjects
            \ProjectName
                    \Data
                    \SqlServer
```

NOTE: If you have more than 1 namespace in a small project and you are only creating one assembly file then name the assembly as per the ROOT Namespace. In the above example this would be: IUProjects.ProjectName

Function Names

public and **protected** methods are Pascal Cased

private functions are Camel Cased

Name the function as per its functionality and not as per any specific algorithm or access method. Do not use Underscores in function names.

Good Example:

```
public bool UpdateClient();
private bool updateClientsAddress();
protected bool UpdateClient();
```

Bad Example:

```
public bool UpdateClientDataset()
private bool updateClientsAddressArray();
public bool Update_Client()
```

Constant Names

Constant names follow the rules for member names so that they can be easily identified. Begin each name with: "**Const**." Note that Private Constants may be all UPPERCASE and contain underscores if desired.

Example:

```
public  const   double ConstPI          = 3.1415;
public  const   string ConstFirstName   = "Bob";
private const   string LAST_NAME        = "Jones";
```

Public and Protected Class Variables

Use Pascal Casing

Example:

```
public      int   AmountOfLoan   = 0;
private     int   MaxSpeed       = 0;
protected   int   BadLoan        = 999;
```

Private Class Variables

Use Camel Casing and begin with an Underscore.

Example:

```
private     int     _maxSpeed     = 9999;
```

Control Names

Use these prefixes to name controls consistently:

⇒ **Table XVI Control Names**

Control	Prefix
Button	cmd
Calendar	cal
CheckBox	chk
CheckBoxList	cbl
DataGrid	dg
DataList	dl
DropDownList	cbo
HyperLink	lnk
Image	img
Label	lbl
ListBox	lb
Panel	pnl
RadioButton	opt
Table	tbl
TextBox	txt

Code Structure

The goal is to make the code easy to read and understand.

- Use whitespace to separate sections of code
- Use tabs to indent and align code sections

Example:

```
string   name          = getUserName();
int      count         = getCount( name, cMaxCount );

for ( int i=0;   i < MAX_SZ;   ++i ) {

         count += i;

         doSomeWork( count );
}
```

Finalizers

Implement IDisposable and/or a Finalizer according to the following:

⇒ **Table XVII Implementing Finalizers**

Your class holds onto	IDisposable	Finalizer
Only **managed** resources that do not implement IDisposable or have any way of being closed	NO	NO
Only managed resources, but some implement IDisposable or can be closed in some way	YES	NO
Both managed and unmanaged resources	Yes	Yes
Only unmanaged resources	Yes	Yes

Finalizers place a heavy burden on the Garbage Collector and therefore should be avoided if possible. Only implement a Finalizer when you are holding onto *Unmanaged* resources that need cleaning up. Any class that implements a Finalizer should also implement the IDisposable interface to announce to the world that users of this class should explicitly call the dispose method when they are done. **When coding a Finalizer, always use the accepted Finalizer pattern.** See the IDisposable topic in the Visual Studio .NET MSDN Help file for more information.

Wrap Finally Clauses that Restore Security-Related State in an Outer Try Block

Finally clauses that restore security-related state will be wrapped in an outer try block to prevent an exception filter further up the stack from executing before a secure environment can be restored. If sensitive operations such as impersonation occur in the try block, and an exception is thrown, the filter can execute before the finally block. For the impersonation example, this means that the filter would execute as the impersonated user. ***Filters are currently implemented only in Visual Basic***.

```
try {
  // Do some work.
  Impersonator imp = new Impersonator(John Doe);
  imp.AddToCreditCardBalance(100);

} finally {
  // Reset security state.
  imp.Revert();
}
```

The following psuedo-code shows the pattern you can use to protect your code.

```
    try {
      // Do some work.
    } finally {
      // Reset security state.
    }

} catch() {
   throw;
}
```

LinkDemand Security Checks on Types do not Protect Access to the Type's Fields

Use public properties NOT public fields. Fields must be secured with a security check other than LinkDemand.

```
// This code requires immediate callers to have full trust.
[System.Security.Permissions.PermissionSetAttribute(System.Security.Permissio
ns.SecurityAction.LinkDemand, Name="FullTrust")]

public class SecuredTypeWithFields  {
     // Even though the type is secured, these fields are not.
     public double xValue;
     public double yValue;

     public SecuredTypeWithFields (double x, double y)
     {
       xValue = x;
       yValue = y;
     }
     public override string ToString()
     {
                 some code….
     }
  }
```

A caller with full trust may create an instance of this type and return it to another caller that does not have full trust. In this case the first caller without full trust may access the public fields but cannot access the public methods.

Short Circuiting in VB

If desired, in VB you may use AndAlso - OrElse to short circuit evaluation of an expression.

Do not use this feature if one of the expressions is a function call, as it may behave as expected.

For example:

```
        If ( x AndAlso y > 9)  then Ok()

‘ Func() will never be called if x is False
        If ( x AndAlso Func(12) > 0 ) then BAD()
```

Saving State

Page and object state can be saved on the Server and/or on the Client. These choices are made depending on the load expected on the server, the amount of memory on the server, security, scalability and more. Each method has its trade offs.

View State

May be encrypted and signed for more secure data transfer (requires all servers in a farm to contain the same encryption configuration setting) This puts the state on the client machine and does not over burden the servers memory BUT it does increase the page size, lowers the throughput and puts more strain on network resources across possibly many network hops.

- Do not use view state, except for small amounts of data.
- Turn off View State in all controls that do not require them (e.g. Labels)

NOTE: if you need to capture the OnChange Event for a control leave it’s ViewState ON

Application Cache / Application Object

1. Use this for data that is shared by all users of the application.
2. If the data is static use the Application Object.
3. If the data changes often then use the Application Cache.
4. Use your judgment as to how much data should be held here as this does use up memory on the server.

Session State

This should be used only for very small pieces of data such as boolean flags or ID numbers. Anything larger should use other methods to save memory. On a web farm this also requires using SQL Server State Service.

SQL Server / NT State Service

These methods must be used to when using session state on a web farm, else the user may be load balanced to a different server and lose any session state data.

1. State Service: leaves DB Server more available, much faster, lose data on re-boot
2. SQL Server: extra load on DB, slower, retains data after re-boot so more secure

Base Page Class

Always use a base page that inherits from the .NET Page class. All other pages in the application should inherit from this base page. This will save duplicating functionality on each page and provide a more robust and traceable application.

The base page should centralize all common functionality needed, such as:
1. Exception Handling
2. Page Navigation / Passing and receiving data between pages
3. Encryption / Decryption
4. Security checks

Isolate Dependencies

Use wrapper methods for code that depends upon special algorithms, storage mechanisms, etc. to allow ease of modification. This would include such things as saving state.

For example rather than doing this:

```
        SomeFunction()
                Session[  DATA_KEY ] = dsClients;   // save a dataset in session
```

Do this:

```
        SomeFunction()
                saveState( dsClients, DATA_KEY )

        saveState( object data, string key )
```

```
                Session[key] = data

        object restoreState( string key )
                return Session[key];
```

This will allow you to change where you are saving state without making modifications throughout the code. It also allows you to add specific error handling and any pre and/or post processing in one place.

List Management

Three controls are provided to manage lists on a web form.

- DataGrid
- DataList
- Repeater

Each control has its pros and cons related to built-in features and speed and scalability. The following is recommended based on test results:

- DataList
  - Use this as the 1st choice UNLESS it does not offer functionality needed
- DataGrid
  - Use only if cannot use the DataList because of missing functionality
- Repeater
  - Use when the DataList is too slow for your needs

Test Results for Binding 100 records from a database:

	#Requests per second supported	
DataGrid	12	(slowest but most functionality)
DataLIst	17	(slower but more functionality than repeater)
Repeater	22	(fastest but not much functionality)

Data Binding

Three ways to bind data to a control exist:

1. Inline formatters	(fastest)
2. Member Methods	(slower but not by much)
3. Event Handlers	(slowest)

It is recommended to use Member Methods for the following reasons:

1. Can format data using code rather than be limited to format expressions only
2. Can compare other elements or form controls to change rules on the fly
3. Is basically cleaner as all code is in code behind

Usage:

In the Item Template section of the control

```
        <ItemTemplate>
        <td><% #ShowData( Container.DataItem, "ClientName"  %></td>
        <td><% #ShowData( Container.DataItem, "ClientPhone"  %></td>
```

In the code behind for the Page

```
        Public string ShowData( object data, string columnName ) {
If ( columnName == "ClientName")   return data[ columnName
].ToString().ToUpper();
                        etc.  for each field format as desired
```

Web Services

Chunky Calls

Use chunky calls to remote objects. This lowers the number of calls that need to be made and improves throughput and uses less server resources.

Example:

```
        Non-Chunky calls:
                Obj.SetName( userName );
                Obj.SetAddress( address );
                Obj.SetPhone( phone );

        Instead use:
                Obj.SetData( username, address, phone );
```

Asynchronous Calls

If possible use Asynchronous Calls to allow the UI to continue working, especially if the called function does not return any data.

MarshallByRef

If possible, always marshal by reference.

Distributing Type Information to the Client

There are three ways to distribute type data to the client for MarshallByRef projects.

1. Simply copy the whole Assembly to the client
    a. Not recommended
    b. Puts full source code on client
    c. Harder to tell if remoting is setup correctly (client will use the local assembly even if setup is wrong)
2. Code to Interfaces and distribute the interface to the client
    a. No source code on client
    b. Requires coding to interface, so may need a Factory object to create the object on the server or use the Activator class to create objects.
    c. Distribute interface to clients
3. Create a meta-data assembly using the soapsuds tool and distribute this assembly to the client
    a. No source code on client
    b. Best choice if not already coding to interfaces

Web Service Proxy Creation

Using VS.NET to create the proxy will hardcode the referenced URL into the proxy. Instead use the WSDL .exe tool to create a proxy that will read the URL from web.config as:

```
Wsdl.exe /urlkey:

        UrlRoot baseurl:http//localhost/
        http://localhost/web/service.asmx?WSDL
```

Web.Config entry in <appSettings> section as:

```
        <add key=”UrlRoot”  value=”http://localhost/” />
```

Change the value of this key to point to another server as needed. Use the IDE to create the reference when developing BUT drop the reference and create a new one using wsdl when ready to deploy to QA or Production.

Note: Type WSDL at the .NET command prompt to get help on available options.

Caching

If possible, Cache any data that requires time and resources to re-create.

Security

Passwords / Connection Strings / Sensitive Data

1. Always encrypt sensitive data.
2. Use aspnet_setreg.exe or the DPAPI to encrypt any sensitive data in the web config file
3. Lock an account after x number of failed logons
4. Restrict Concurrent logons (re-logon to a session from a different IP address)

Database Accounts

1. Never use an administrative account to log onto the database.
2. If not using ACL's and windows log on in an intranet situation then always create a new service account with the minimum rights needed by the application.
3. A service account will also allow all users to share the same logon and take advantage of connection pooling.
4. The account will have only execute rights to the stored procedures and no base table access.

Code Security

1. Always use CAS to DEMAND that the caller has permissions to access system resources.
2. Use an Obfuscation tool to encode assemblies before release to production

Web Config Security Settings

1. Use URL Authorization to Allow/Deny specific users or roles
2. Use the CustomErrors section to define a default Error Page
3. Set the MachineKey element to the same value for all web servers in a web farm

Note: Use aspnet_setreg.exe or the DPAPI to encrypt any sensitive data in the web config file

Security Testing and Tools

1. Security test all applications before deployment to production
2. Test the apps configuration for weaknesses
3. Use the security tools provided by Microsoft (e.g. FxCop) to help find problems

Security Tools: http://msdn.microsoft.com/security/downloads/tools/default.aspx

Isolated Storage

This is NOT a secure storage area and should NOT be used to store sensitive data that is not encrypted.

Component Deployment

1. Place all sensitive code and data (Business and DAL) behind a firewall, accessible only by known and trusted accounts
2. UI Layer in the DMZ (**No sensitive data or functionality in this layer**)
3. Business and Data layers behind a firewall, using IPSec to allow only access from the DMZ Server. (OR can also use SSL with code access security and strong names to demand caller has the correct permissions and/or is a known application)

Security Policy

Be aware that the security policy setup by the network administrators will affect all code running on the network. You must coordinate any special permission your app requires with this policy.

Configuration Files

Reading Data

1. Reading data from the web.config file must be done so as not to create any un-handled exceptions at runtime because of missing data.

2. Always read the data and check for null values or always supply an empty string as:

```
String connect = AppSetting[  “DB_CONNECT” ]  +  “”;
```

3. Create a function that is called at startup to read and check all setting that the application cannot do without, throw a custom exception (display only non-sensitive data to user) if any data is missing

```
private void CheckConfigValues() {
String connect = AppSetting[  “DB_CONNECT” ]  +  “”;
If  (  connect.Trim().Length  ==  0  )  throw  new
CustomException(“Missing Connection info”);
```

Development Settings vs. Production Settings

To allow the web.config file to hold production setting and still use development settings during development without changing the web.config use the FILE attribute to point to a file that holds only development settings.

Example:

Web.config

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
        <appSettings
file=”devsetting.config”>
            <add  Key=”Connect”
value=”Production Settings here” >
```

devsettings.config

```
<appSettings>
    <add Key=”Connect”  value=”Development Settings here”
>
```

NOTE: the devsetting.config file only has the <appSettings> section, no other sections are needed or allowed. At runtime ASP.NET will see the FILE attribute in web.config and will look for an existing devsetting.config file to read the data from. If the file does NOT exist data will be read from the web.config file. When deploying to production do not deploy the devsetting.config file

NOTE: Changes to the devsettings.config will NOT cause a ASP.NET to recompile the application and your new setting will NOT be picked up unless you restart IIS –or- modify a line in the web.config file as:

```
<add Key=”Dummy”   value=”Modified 10-10-2003” >
```

just change the date to get ASP.NET to pick up the changes

ASP.NET Worker Process Identity

The asp.net worker process runs with limited permissions to protect the system. *DO NOT change the identity in machine.config. This is a job for a system administrator only. It affects all applications on the machine.*

User Interface Layer

General

Authentication

Authenticate users in this layer and if needed pass to Business Layer with a custom principal object or other mechanism as per the Hosting application and transport being used (e.g. channel sinks, soap header).

Exception Handling

Provide a method to handle all un-handled exceptions in the UI Layer and all other layers.

Business Emissary

To avoid cross-app domain calls simple business logic like formatting and user input validation may be moved into a separate class in the UI layer known as a Business Emissary. This will keep all this type of logic in one central location in the UI layer and avoid expensive calls across app domains.

This emissary class should be the only gateway to the Business component. All calls from the UI to the business layer should go through this class. This class either handles the request directly or it calls into the business layer directly (to get the requested data, or perform a requested operation).

This emissary class may know a little about both the UI and Business layers. For instance, functions that map database columns to form controls may be placed here.

NOTE: No sensitive information should be kept in this emissary class, unless encrypted.

Business Layer

General

Exception Handling

Trap All Exceptions. Bubble Up any un-handled exceptions to the UI layer.

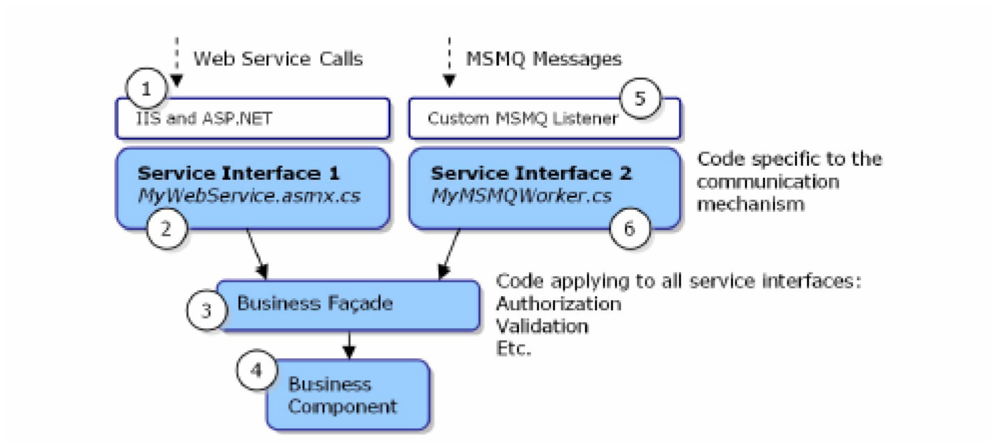
Service Interface

If this component is used by many applications you may want to provide a service interface layer to enable upgrades and modifications without breaking existing code. See below for more.

Service Interfaces

A Service Interface provides a way for clients to call into an application service indirectly. This interface provides:

1. Shields the client from any changes made to the interface.
2. Allows a central place to call many services from
3. Central place to perform any logging, auditing, and authentication needed.
4. A façade to implement business tasks
   a. User may make 1 call to a function that in turn may call to 1 or more actual services to complete the process as per the business rules.
   b. To expose the same services to different callers that may require different authentication or SLA requirements
   c. To provide a central place to deal with possibly different communication channels used by different clients (e.g. soap, MSMQ, binary TCP)



Passing Data to the Façade

Selecting a data type to use for passing arguments to a service interface depends on too many variables to define a standard. The following are some suggestions:

1. Dataset
    a. Create a row(s) of data to hold the parameters being passed
    b. Use the schema to provide information on the data type of the parameter
    c. You may also choose to return any values to the client in a dataset as well
2. ArrayList / SortedList
    a. Each element can hold any object type
    b. Use a Name Value collection to hold parameter names and values

Transaction Management

Your service interface will need to deal with a channel that provides transactional capabilities (such as Message Queuing) or one that doesn't (such as XML Web services). It is very important that you design your transaction boundaries so that operations can be retried in face of an error. To do so, make sure that all the resources you use are transactional, mark your root component as "requires transaction," and mark all sub components as either "requires transaction" or "supports transactions."

With transactional messaging mechanisms, the service interface starts the transaction first and then picks up the message. If the transaction rolls back, the message is automatically "unreceived" and is placed back in the queue for a retry. When using Message Queuing, Enterprise Services Queued Components, or Message Queuing Triggers, you can define a message queue-and-receive operation as transactional to achieve this automatically.

If you are using a messaging mechanism that is not transactional (such as XML Web services), you need to call the root of the transaction from the code in the service interface. In the case of a failure, you can design the service interface code to retry the operation or return to the caller an appropriate exception or preset data representing a failure.

ASP.NET Performance

This section will describe ways to increase the performance of your asp.net application.

View State / Session State / Application Cache / Application object

1. Turn OFF **Viewstate** in any controls that do not need it (e.g. labels, datagrids that are always re-filled on postbacks anyway, etc.)

2. Store only Value Types in **Viewstate** , **Session State, Cache, and Application**. Or store Reference Types that are Serializable only.

   a. Add the [Serializable] attribute to any custom objects being stored

3. Avoid use of the **Application** object as it requires locking and unlocking to access objects. Use the **Cache** instead.

4. **Cache** Callbacks – DO NOT create a callback function for the cache in the Page class. This will cause the page (fro each user) to remain in memory for the duration of the cache. Instead create a separate class with a static (shared in VB) callback function. This can be shared by all pages in the application and will avoid a build up of pages in memory.

5. Another option for storing Application level data is to use the **Global.asax** file. Create a static (shared in VB) variable to hold your data and fill it in the Application_OnLoad event. This keeps the data in memory and allows for faster access. (Note: balance this against the size of the data being stored)

Page Event Processing / Page Directives

1. Avoid creating custom events for controls

2. **Page_Load**: Always check IsPostBack in the Page_Load event, to avoid any duplicate processing of data. For example, if you load a dropdown list from a database, this list will be saved in viewstate, so there would be no need to re-query or re-fill the list on Post Backs

3. **Page_PreRender**: This is the best place to bind any data to avoid binding more than once (e.g. in Page_Load and again in the

ItemCommand event of a datagrid, which both fire on a postback event from the datagrid)

4. Eliminate all possible processing that is not needed in event code. (e.g. use a separate server side process to handle computations etc. that are not needed immediately in event code)

5. **Page_Unload**: use this event to explicitly **DISPOSE** of any resources (close files, connections etc.) All Server Control also implement a dispose method, calling this method explicitly will allow the resources to be cleaned up quicker than waiting for the variable to go out of scope.

6. **Smart Navigation** – set this to true in the page directive. Although this will only achieve gains in IE 5.0+ browsers it will have no bad side affect in other browsers.

    a. <@Page Language="vb" smartNavigation="true" %>

Perceived Performance

1. **Page Loader**: Use a Page Loader for any pages that take a long time to load. This will allow the user to perceive that something is being done rather than nothing is being done.

    a. A sample Page Loader is available in the ITSD samples in source safe database: **DemoPageLoader – you can include this page in your app as needed. See the source code for details**

2. **Background Processing and IFrames :** If you have a page that contains some static data as well as dynamic content and the dynamic content takes a while to load then: Use IFrames to load the dynamic content as:

    a. <iframe id="frmDetails" style=".." src="details.aspx"> </iframe>

    b. You may also include a call in the iframe to the PageLoader above so that the user will see something happening while it is loading.

3. **Page Caching**: Cache dynamic pages that can possibly be made "static" for a duration of time. For example a page that hits the database for a list that will probably not change for at least 1 hour can be cached for 1 hour to avoid hitting the database again as:

    <%@ OutputCache Duration="3600" VaryByParam="none">

Note: see VaryByParam in MSDN Help for more options

4. **Partial Page Caching:** Use partial page caching for Menus, Sidebars etc. that are static in nature. Create User Controls to hold these items for page caching.

   NOTE: *never try to access a user control (that is cached) in your code. The only safe access is in the user control code itself. This is usually not a problem as you will only need to respond to the events the control produces anyway.*

5. **URL's**

   a. Use Relative URL's to transfer control to pages in your app

   b. Use an ending backslash "/" when requesting a default page (when not including the page name in the URL) . This will save a roundtrip from the server to the browser.

Exception Handling

Avoid Try Catch statements that are not needed. Do not depend on throwing an exception to catch a possibly known error. Test for any known errors in code. For example the following code is a bad way to determine a value of zero

```
Try {
    int x = y/z;      return x;
} catch(DivideByZeroException ex) {
    return 0;
}
INSTEAD use:
    if ( z != 0 )     return y/z;
    else              return 0;
```

NOTE: This does not imply that you should not use exception handling. Just be aware that it consumes more resources than a simple check you can do yourself for know issues like null pointers, divide by zero etc.

Web Services

1. Execute web services **asynchronously**. As the web service automatically provides methods to accomplish this it is easy to

implement and will allow your app to not hang while waiting for a slow connection or service or failed connection etc.

2. Cache web service return values that do not change over a short period of time to avoid calling the service again. This can be done using the Application Cache or by setting it in the web method itself as:

    a. <WebMethod(CacheDuration:=120> Public Function GetData()

3. Set the Timeout property to insure you do not wait too long for a web service to return data.

    ```
    Dim obj as New localhost.ServiceName
    Obj.Timeout = 10000
    ```

4. Loading Images – Example: you have a datagrid that is filled with some information and for each row in the grid an image file is also required and the image is retrieved from a web service.

    a. Add a template column in the grid for the image and in the template add the following code:

        ```
        <img src=”LocalDefaultImage.gif’
        Onerror=”javascript:this.src=’LocalDefaultImage.gif’”
        Onload=”javascript:this.src=’GetImage.aspx?id=<%#Conta
        iner.DataItem(“id”)%>’;”>
        ```

        Where GetImage.aspx calls into the web service to retrieve the image. This allows the image to be loaded in the background while the user can view the other data in the grid. LocalDefaultImage is a small gif file in the local directory that acts as a placeholder while the image is being loaded.

Code Optimizations

1. **Early Binding In VB**: use Option Strict so as to force declaration of types for all variables. Late binding occurs when a variable is declared as: dim s, the runtime must then use reflection to get the correct type. Always declare variables as: dim s as String

2. **Evalutaions**: use AndAlso / OrElse constructs to short circuit evaluations

3. **Boxing**: Avoid boxing when possible. For example if you know the number of items you are saving use an array of the type rather than an ArrayList. If you must use an ArrayList then use the constructor to initialize a starting size as: new ArrayList(1000) to improve performance somewhat.

4. **Strings** – If doing many concatenations on a string use the StringBuilder object instead of a String object (which creates a new string each time)

5. **SQL Command Builder** – Avoid using the command builder for production applications. This requires extra queries against the database to create the commands. Use them for testing only then create your own sql manually for production.

Section IX SQL Best Practices

This section provides a framework to aid in optimal usability of the Microsoft SQL Server 2000 schema, scripts and stored procedures developed for applications by defining a reasonable, consistent and effective coding style.

This framework serves to improve the application without unnecessary impact on development and unnecessary controls on personal coding preferences. For these reasons the framework focuses on identifier naming conventions that are intended to be used by all developers, general style guidelines indicating the preferred format and usage of SQL language components, and a definition of the database development methodology.

Identifiers

This framework focuses on identifier naming conventions that are a preferred format by ITSD SQL Server Unit and are intended to be used as a best practice in coding.

Case

- Use all upper case for table and view names
- Use mixed case for column names and variables
- Use mixed case for stored procedure name
- Use lower case for other names except use the same case as indicated above where a table or column is used in another object's name

Prefixes and Suffixes

Use the following standards for prefixes and suffixes.

Database Objects

- Use the following standard prefixes for database objects:

Object type	Prefix	Example
Primary key Clustered	`pkc_`	`pkc_MY_TABLE__Column`
Primary key Nonclustered	`pkn_`	`pkn_TB_TABLE__Column_List`
Index Clustered	`ixc_`	`ixc_TS2_TABLE__Column`
Index Nonclustered	`ixn_`	`ixn_TB_TABLE__Column_List`
Foreign key	`fk_`	`fk_THIS_TABLE__ColumnB__to__TB_PKEY_TABLE__ColumnA`
Unique Constraint	`unq_`	`unq_TB_TABLE__Column_List`
Check Constraint	`chk_`	`chk_TB_TABLE__Column`
Column Default	`dft_`	`dft_TB_TABLE_Column_List`
Passed Parameter	`@p`	`@pPassedVariableName`
Local Variable	`@`	`@VariableName`
Table	`TB_, *_`	`TB_TABLE_NAME` (see detail below)
View	`VW_`	`VW_NET_ACTIVE_UNITS`
User Defined Scalar Function	`ufs_`	`ufs_return_value_name`
User Defined Table Function	`uft_`	`uft_TB_TABLE_NAME`
Stored Procedure	`usp_`	`Eds_Def (Note: Stored Procedures CANNOT be prefaced with SP_ as these have special meaning and execution rules in SQL server). Example: usp_search_county`

Scripts

- Use the following standard prefixes for scripts:

Script type	Prefix	Example
Stored procedure script	`PROCEDURE_`	`PROCEDURE_Calendar.sql`
Schema script	`SCHEMA_`	`SCHEMA_Calender.sql`
Conversion script	`CONVERSION_`	`CONVERSION_Schedule.sql`
Rollback script	`ROLLBACK_`	`ROLLBACK_Schedule.sql`

- Save all scripts using the .sql extension
- All other suffixes should be whole words

  Name, Type, Flag, etc. …

Stored Procedures

- Stored procedure names ought to reflect the name of the primary data source, the action the procedure accomplishes, and the audience that uses the procedure.
  - Generally, the best table name is the one that represents most of the data or the primary join table.
  - The audience might typically be one of the following:
    - Import
    - Export
    - Custom
    - Operations
    - Report
    - System (not directly available to the front end)
- Do not create a user stored procedure with the “SP_” prefix. This has a specific meaning in SQL Server and could negatively impact performance.
- Avoid abbreviations where possible, although acronyms that have meaning to the business are acceptable.

Tables

Create Table

- All objects in the production database are owned by (DBO). Developers should not create objects own by his/her accounts.
- Add the primary key as a first element (or elements if a composite key) when creating a table
  - In many cases it is easier to script the table in multiple steps. The first script creates the table and columns. The second script adds the indexes and constraints. This is not only easier to read in some

cases but also helps if you have to drop indexes on a table to do a fast BCP operation etc…(because you already have the script isolated and identified).

- Object names should include underscores between words.
- Name all tables in the singular form.

```
TB_CALLING_CARD not TB_CALLING_CARDS
```

- If a column references an **Id** in another table, use the full table name. For example, use **Title_Id** in table **TB_AUTHOR** to reference column **Id** or **Title_Id** in table **TB_TITLE**.
- Explicitly name constraints. A table or column constraint name will include the table name(s) that it references and the columns affected by the constraint. Separate each table and column in a constraint name with two underscores (__) to differentiate from the single underscore (_) that may be within the table or column's name.
- A foreign key name will identify both tables participating in a foreign key, the column(s) involved in the relationship, and the direction. The foreign key table (table where constraint is attached) appears first.

```
fk_TB_COURSE__Educator_Id__to__TB_USER__Id
```

- Default constraints must be defined at the column level.
- Define all Constraints other than defaults at the table level.
- Avoid rules, database level defaults that must be bound, or user defined data types. While these are legitimate database constructs, opt for constraints and column defaults to hold the database consistent for development and conversion coding.
- Never use a SQL Server reserved word as an identifier name. (Refer to SQL Server Books on Line for a complete listing).

Altering Tables

- **alter table** should be used in scripts that will be used for upgrading existing table, not when creating tables.
- Use the following outline to drop an existing column or table constraint:

```
          If (objectProperty(object_id('{constraint
     name}'),
                                              'IsConstraint') is
                                         not null)
                   alter table {table name}
                         drop constraint {constraint name}
```

- Use the following outline for adding or changing an existing column or table constraint.

Example

```
          If (objectProperty(object_id('{constraint name}'),
                                         'IsConstraint') is
                                  null)
                alter table {fully qualified table name}
                      add constraint {constraint name}
                            default {constraint value}
                    [for {column name}]
```

- Use the following outline to drop an existing column.

Example

```
       If (ColumnProperty(object_id('{table name}')
                          ,{column name},
                          'AllowsNull') is not null)
                alter table {fully qualified table name}
                      drop {column name}
```

- Use the following outline for adding a column.

Example

```
       If (ColumnProperty(object_id('{table name}')
                          ,{column name},
                          'AllowsNull') is null)
                alter table {fully qualified table name}
                add {column name} {data type} {null | not
          null}
                      [constraint {default name}
                            default ({default value})]
```

Indexes

- Use the following outline for creating Indexes

Example

```
            create {clustered | nonclustered} index {index name}
                  on {fully qualified table name}
                  ({column list})
                  {options}
```

```
            create nonclustered index ixn_TB_TICKET__Expire_Dt
                  on Events.dbo.TB_EVENT(Expire_Dt)
```

  - Explicitly name all indexes and include the table name and all indexed columns in index order.

    ix{c | n}_{table name}__{column name}[__{column_name}[..]]

Example

```
            ixn_TB_DISTRIBUTOR_Name
            ixc_TB_ACTIVITY__Itinirary_Id__Active_Dt
            ixn_TB_COURSE__Instructor_Id
```

Stored Procedures (and other DML scripts)

- Use the following outline for creating stored procedures

Example

```
use {database name}
if (objectProperty(object_id('{owner}.{procedure name}'),
                          'IsPRocedure') is not null)
   drop procedure {owner}.{procedure name}

GO

create procedure {owner}.{procedure name}
   [{parameter}  {data type}][,
      …]
as
/*******************************************************************
```

```
* PROCEDURE: {procedure name}
   * PURPOSE: {brief procedure description}
   * NOTES: {special set up or requirements, etc.}
   * CREATED:  {developer name} {date}
   * MODIFIED
   * DATE            AUTHOR                  DESCRIPTION
*-------------------------------------------------------------------
   * {date}          {developer} {brief modification description}
   *************************************************************/
[declare {variable name} {data type}[,
…]]

[{set session}]

[{initialize variables}]

{body of procedure}

return

{error handler}
```

- The owner of stored procedures should always be DBO to prevent broken ownership chains.
- Do not use temporary stored procedures.
- Do not define default values for parameters. If a default is needed, the front end will supply the value.
- Do not use output parameters. Any information returned to the client should be done via a result set.
- Do not create stored procedures that return multiple result sets.
- Do not use the encryption option except as other wise noted above.

Note: SQL Server Unit requires to have access to the full source of the stored procedures.

- Never use the recompile option in stored procedures.
- Place all declare statements before any other code in the procedure to give the query optimizer the best shot at reusing query plans.
- Place set statements before any executing code in the procedure.
- Avoid calling stored procedures from stored procedures because error handling is very unpredictable in such circumstances.
- Fully qualify all system stored procedures used in a stored procedure. This will optimize performance as the system will never have to search for the system procedure.
- Enumerate all column lists. Do not use the '*' wildcard.

- In order to capture two or more SQL Server global variables pertaining to the same statement, declare a variable for each global to be identified and assign all in a single select immediately after the statement. Capturing more than one global variable one at a time will produce erroneous results.

Example

```
Declare @iRowCount int,
      @iError
Select Id from MyDB.dbo.MyTable
Select @iRowcount = @@rowcount,
      @iError = @@error
if @iError <> 0
      goto MyErrorHandler
if @iRowCount > 0
      delete MyDB.dbo.MyTable
```

Not

```
select Id from MyDB.dbo.MyTable
if @@error <> 0
      goto MyErrorHandler
if @@rowcount > 0 -- rowcount will be
about if @@error
      delete MyDB.dbo.MyTable
```

Error Handling

- Avoid abbreviations other than the specified prefixes and postfixes in error messages.
- Use system messages stored in syscomments. Use the following outline for messages:

`{fully qualified procedure name} : {message}`

Example

```
      MyDatabase.dbo.MyStoredProcedure : A strange error has
occurred?
```

- Error messages should be added to the system using the following outline: Developers will be required to provide this information to the SQL Server Unit. If the application will be hosted on a system with other applications, please consult with the SQL Server Unit for a unique message id.

Example

```
      sp_addmessage      msg_id,
                         severity,
                         {message text}[,
                         {language}[,
                         'with_log'[,
                         'replace']]]
```

- Capture the fully qualified procedure name by inserting the following code once at the beginning of each procedure that might raise the error:

Example

```
            declare @sProcedureName varchar(255)
            select @sProcedureName = db_name()
                        + '.'
                        +
                        user_name(objectproperty(@@procid,'OwnerI
                        d')) + '.'
                        + object_name(@@procid)
```

- Assign error message numbers based in the following

Example

○ Reserved	50000 thru 50999
○ General errors	51000 thru 51099
○ Import errors	52000 thru 52099
○ Export errors	53000 thru 53099
○ Services errors	58000 thru 58099
○ DbChangeControl Message	59000 thru 59099

Schema Scripts

- **`go`** is the standard TSQL batch separator. Do not write scripts that depend on another separator. **Go** should appear on its own line of the script.
- Each **`def_`** script should define only one table. The **`def_`** includes all constraints, keys, and indexes for the table. A **`def_`** script only creates an object. A **`def_`** script must not drop statements.
- **`Proc_`** and **`conv_`** scripts should be able to be run multiple times against the same database with no adverse effect and no errors. Always check for the existence of each object before creating it again to avoid meaningless errors in the scripts output stream.

- An **`rbk_`** script must accompany every **`conv_`** script. As with the **`conv_`** script, the **`rbk_`** script must be written so that it will execute multiple times with no adverse effects and no errors. The **`rbk_`** script will reverse or rollback all changes applied in the **`conv_`** such that full *a priori* functionality exists in all databases touched by the **`conv_`** script.

Formatting

- Use single quote characters to delimit strings. Nest single quotes to express a single quote or apostrophe within a string

```
set @sExample = ‘Bill’’s example’
```

- Use parenthesis to increase readability, especially when working with branch conditions or complicated expressions.

```
if ((select 1 where 1 = 2) is not null)
```

- Use **`begin..end`** blocks only when multiple statements are present within a conditional code segment.
- Limit the length of lines in all source code to 114 characters. If possible, try to leave all code viewable without the need to horizontally scroll an 800 x 600 IDE window using a 12 pitch Courier New font – 100 characters.
- Indent one tab when indentation is required.

Whitespace

- Use one blank line to separate code sections.
- Do not use white space in identifiers

DML Statements (select, insert, update, delete)

- Use ANSI join syntax

```
select c.Name, a.Description
from User.dbo.TB_ADDRESS a
inner join VIOLATIONS.dbo.TB_INCIDENT i
On a.Id = i.Address_Id
```

- Use ANSI operators

```
=, >, <, <>, in, exists, not, like, is null, and, or
```

- A correlated sub-query **exists** or **not exists** is preferred over the equivalent **in** or **not in** sub-query.

- Avoid the use of cross joins if possible.

- When a result set is not needed, use syntax that does not return a result set.

```
If exists(select 1
            from County.dbo.TB_LOCATION
            where Type = 50)
      rather than,
if ((select count(Id)
      from county.dbo.TB_LOCATION
      where Type = 50) > 0)
```

- If more than one table is involved in a **from** clause, each column name must be qualified using either the complete table name or an alias. The alias is preferred.
- Do not use the **identitycol** or **rowguidcol**
- Always use column names in an order by clause. Avoid positional references.

Select

- Do not use a select statement to create a new table (by supplying an **into** table that does not **exist)**.
- When returning a variable or computed expression, always supply a friendly alias to the client.

```
select @@identity as Exam_Id,
      (@pointsReceived / @pTotalPoints) as Average
```

- Opt for more descriptive alias.

```
      select @@identity as UserId
      is preferred over
      select @@identity as Id
```

- Use the following outline for select statements. Each column in the select list should appear on its own line. Each unrelated constraint within the where clause should appear on its own line.

Example

```
select {[alias.]column name}[,
             {[alias.]column name}[,
             …]]
      from {database name}.{object owner}.{table name} [[{alias 1}]
      [inner join {database name}.{owner name}.{table name} [{alias
      2}]
      on {alias 1}.{column name} = {alias 2}.{column name}[
      {next join}]]
      [where {constraint condition}
      [and {constraint condition}
      […]]]
      [group by {column [list]}
            [having {constraint condition}]]
      [order by {column [list]}]
            [{union}
      {next select statement}]
```

```
            select      t.Task_Id,
                        t.Course_Id,
                        t.Due_Dt,
                        t.Start_Time,
                        t.End_Time,
                        t.Name,
                        et.Completed_Flag,
                        et.Completed_Dt
            from employee.dbo.TB_TASK t
            inner join employee.dbo.ENROLLMENTTASK et
            on t.Task_Id = et.Task_Id
            where t.Due_Dt >= @pStartDate
            and t.Due_Dt <= @pEndDate
                    and et.Member_Id = @pMemberId
                 order by    t.Due_Dt,  t.Start_Time
```

- When dealing with select statements used as conditions or sub-query, use more convenient formatting than the outline above where necessary. One line select statements are fine as long as they are easy enough to read.
- Dependent constraints within the where clause should appear together offset by parenthesis. Use additional indentation if necessary.

Example

```
   select t.TASK_ID
   from Task.dbo.TASK t
      inner join Task.dbo.ENROLLMENT et
      on t.TASK_ID = et.TASK_ID
      where et.MEMBER_ID = @pMemberId
      and ((t.DUE_DT <= @pStartDate)
            or (t.DUE_DT >= @pEndDate)
                   or (et.COMPLETED_FLAG = 1))
```

Inserts

- Always list column names within an insert statement. Never perform inserts based on column position alone.
- Use the following outline for insert statements moving values or variables into a single row. Place each column name and value on its' own line and indent both so they match as shown.

Example

```
   insert [into] {database name}.{owner}.{table name}
                  ({column name}[,
                  {column name}[,
                  …}})
      values
                  ({value or variable}[,  --{comment hard coded
      value}
                  {value or variable}[,   --{comment hard coded
      value}
                       …]]
```

```
      insert Parts.dbo.TB_Inventory
                  (TOASTER_ID,
                  MANUFACTURER_ID,
                  NAME,
                  NOTES)
      values      (1,                           -- example only
                  1,                            -- example only
                  'Blue card',                  -- example only
                  'designer card')              -- example only
```

- Use the following outline for inserts that move data from one table to another. Break and indent the column lists so they match. Apply the same formatting to the **`from`** clause as described in the select statement.

Example

```
   insert [into] {{database name}.{owner}.{table name} | {alias}}
                  ({column name}[,
                  {column name}[,
   …}})
select [{target column name =}]({column name}[,
                  [{target column name =}]{column name}[,
                  …}})
      {from clause
                          [{where clause}]}
```

```
            insert into Parts.dbo.TB_TRACTOR
                        (Tractor_Id,
                        Manufacturer_Id,
                      Name)
    `      select      PartId,
                       @sBoltId,
                       'plow bolts'      -- name from vendor
                 catalog
                  from Equipment.dbo.TB_HEAVYDUTY
                       where Id = @pTractorId
```

- Provide an inline comment to explain any hard-coded value.

Updates

- Use the following outline for simple update statements. Format the where clause as described earlier.

Example

`update {database name}.{owner}.{table name}`
`            set   {column} = {expression}[,`
`                  {column} = {expression}[,`
`                  …]]`
`                           {where clause}`

`            update Articles.dbo.TB_STATISTICS`
`            set  READ_HITS = READ_HITS + 1,`
`                  LAST_READ_DT = current_timestamp`
`            where ARTICLE_ID = @pArticleId`

- Use the following outline for table-to-table update statements. Format the from and where clauses as described earlier.

Example

`   update {database name}.{owner}.{table name}`
`            set   {column} = {expression}[,`
`                  {column} = {expression}[,`
`                  …]]`
`      {from clause}`
`                          [{where clause}]`

`            update PUBS.dbo.TB_TITLES`
`                  set Total_Sales = t.Total_Sales + s.Quantity`
`                  from Pubs.dbo.TB_TITLES t`
`            inner join Pubs.dbo.TB_SALES s`
`            on t.Title_Id = s.Title_Id`

Deletes

- Use the following outline for simple delete statements. Format the where clause as described earlier.

Example

`delete [from] {database name}.{owner}.{table name}`
 `{where clause}`

`delete from WebLog.dbo.TB_ARTICLE_STATISTICS`
 `where ARTICLE_ID = @pArticleId`

- Use the following outline for table-driven delete statements. Use a subquery (formatted as described earlier) rather than using the TSQL extension form.

Example

`delete [from] {database name}.{owner}.{table name}`
 `where [not] exists {correlated subquery`
 `expression}`

`delete WebLog.dbo.TB_ARTICLE_STATISTICS as`
 `where exists (select ID`
 `from`
 `ARTICLES.dbo.TB_EXPIRED`
 `where ARTICLE_ID =`
 `as.ARTICLE_ID)`

Cursors

- Use cursors only where a set based operation is inappropriate. SQL Server Unit will review the cursors usage prior to installation on a shared server.
- Never use a cursor to return data to the application. The performance hit for this is unacceptable.

Team Development Process

The CDHS has adopted a new application hosting model for development and code deployment that is consistent with the technologies supported in our infrastructure. This section describes the development process consistent with the CDHS application architecture and infrastructure standards.

Initiating a New Project in CDHS

ASP.NET applications require a very specific set up process in CDHS. This process requires the coordination of multiple Units and the developers for a typical web based application. Before any code is written the following steps must be taken:

1. The Internet Unit (IU) and SQL Server Unit (SSU) Questionnaire must be completed
2. A meeting with the IU and SSU must take place before any development takes place. This is a very important step because a VSS folder must be created.
3. IU will create a folder structure within VSS for the new application
4. IU will create the web sites and associated web addresses on the various servers
5. SSU creates the needed database for the project
6. SSU supplies the DAL to the developer which should be placed in the developers bin folder of the application server hosting the web services

After the initial setup steps are completed the developers need to create a development environment to match that of the hosting environment on their local workstations. The developers will need to remain in contact with the IU to assure that all the folders are matching across all servers.

Overview of the Team Development Environment

CDHS adopted an **isolated** web application development model. This model allows CDHS to provide a flexible development environment. Using this model, the developer works(edit, debug and run) in complete isolation on a local development workstation using the web server (http://localhost). Access to master source files is controlled via a Visual SourceSafe (VSS) database located on a network file share.

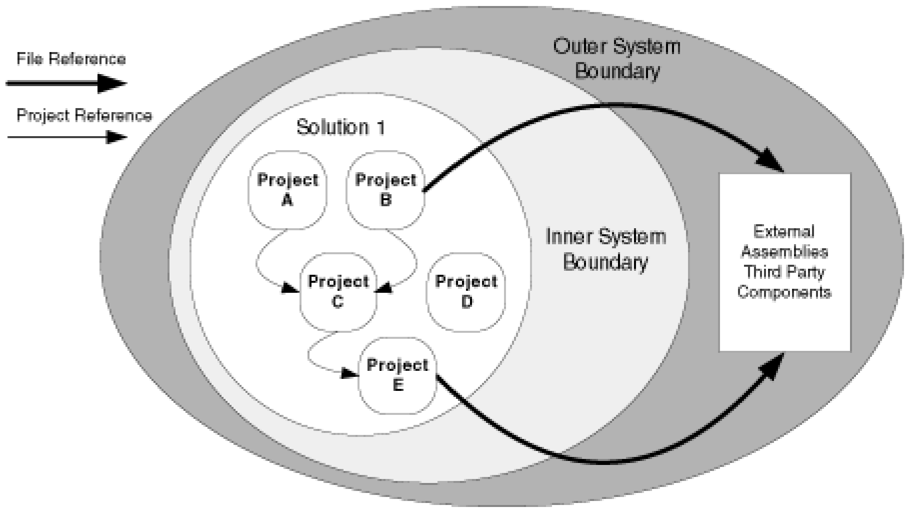
Adopting an isolated development model provides the following advantages:

- Team members develop independently of one another using separate (local) instances of the web application.
- Developers can both develop and debug the application without inadvertently interfering with one another.
- The method provides superior support for source-code control (compared to the non-isolated model that uses FrontPage Extensions).
- The method is slightly faster in a local area network (LAN) environment (compared to FrontPage Extensions).

Partitioning Solutions and Projects when creating new applications is a critical process. CDHS chose to use a **single solution model.** Exceptions may apply for applications with a large number of developers. The partitioning of a solution has a significant impact on the development efforts and build process functions in a team environment. With the single solution model, the developer creates a single Visual Studio.NET solution and uses it as a container for all of the projects defined by the application. Note the following when using a single solution model:

- If one project needs to reference an assembly generated by another, use project references.
- File references should be used only to reference outer-system assemblies (such as .NET Framework assemblies and third-party assemblies) that are not built with the rest of your system.

⇒ **Figure 29 Visual SourceSafe Single Solution Model**



The single solution model offers the following advantages:

- When referencing another assembly generated by a separate project, a project reference is used. Project references are the preferred way to establish references to other assemblies and they are guaranteed to work on all development workstations in a team environment.

- Assembly versioning issues are avoided, because Visual Studio .NET detects when a client of a referenced assembly needs to be rebuilt.
- Project references are sensitive to changes in the configuration of the referenced project meaning that the developer can automatically switch from Debug and Release builds across projects without having to reset references.
- The system build process and build script is much simpler.

The most flexible and least complicated method is adopted by CDHS for its hosted applications. The **isolated** development model allows developers the ability to create, build and test code on their local systems. A **single solution model** gives both developers and CDHS the least complicated model for hosting .NET applications.

Developer Workstation

When creating ASP.NET and web services using Visual SourceSafe (VSS) certain tools and configuration settings need to be applied to the developer's workstation. Contact the Internet Unit for the most current build and maintenance information. The application of IIS to the workstation must be cleared in accordance with Section 6-1030 of the CDHS Information Security Policy prior to contacting the Internet Unit.

Essential Servers

Supporting servers are maintained as a component of the CDHS infrastructure. The following describes the various servers and their configurations, assisting the developer in understanding the development environment.

Visual SourceSafe

This central server hosts one or more Microsoft Visual SourceSafe (VSS) databases used to provide version controlled access to project source files. The developer interacts with this server on a daily basis, checking project files in and out through the Microsoft Visual Studio .NET integrated development environment (IDE).

Web Services Server

The primary function of the Web services server in the team development environment is to host Extensible Markup Language (XML) web services that are currently under development. While the development team members responsible for web services develop them on their local workstations using local instances of Microsoft Internet Information Server (IIS), the central web server allows the services to be published for other developers or development teams to reference from client projects.

Database Server

These servers host instances of Microsoft SQL Server™ and provide a central location to which developers can connect to databases whose schemas match the current system database design. In the new mode, all access to the data is through the DAL.

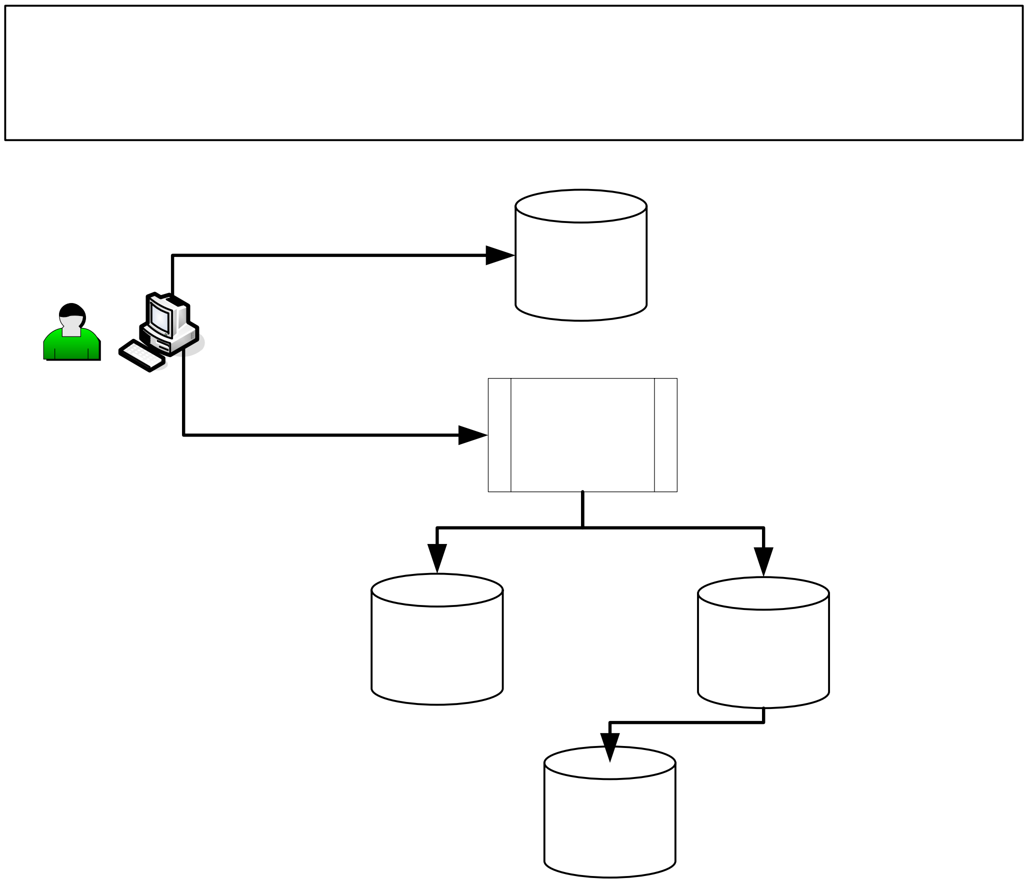
Code Version Control & Promotion

The following sections demonstrate the logical and actual upload and flow process for code version control and code promotion.

Logical View

The following diagram demonstrates the logical development model.

⇒ **Figure 30 Logical Team Development Process**



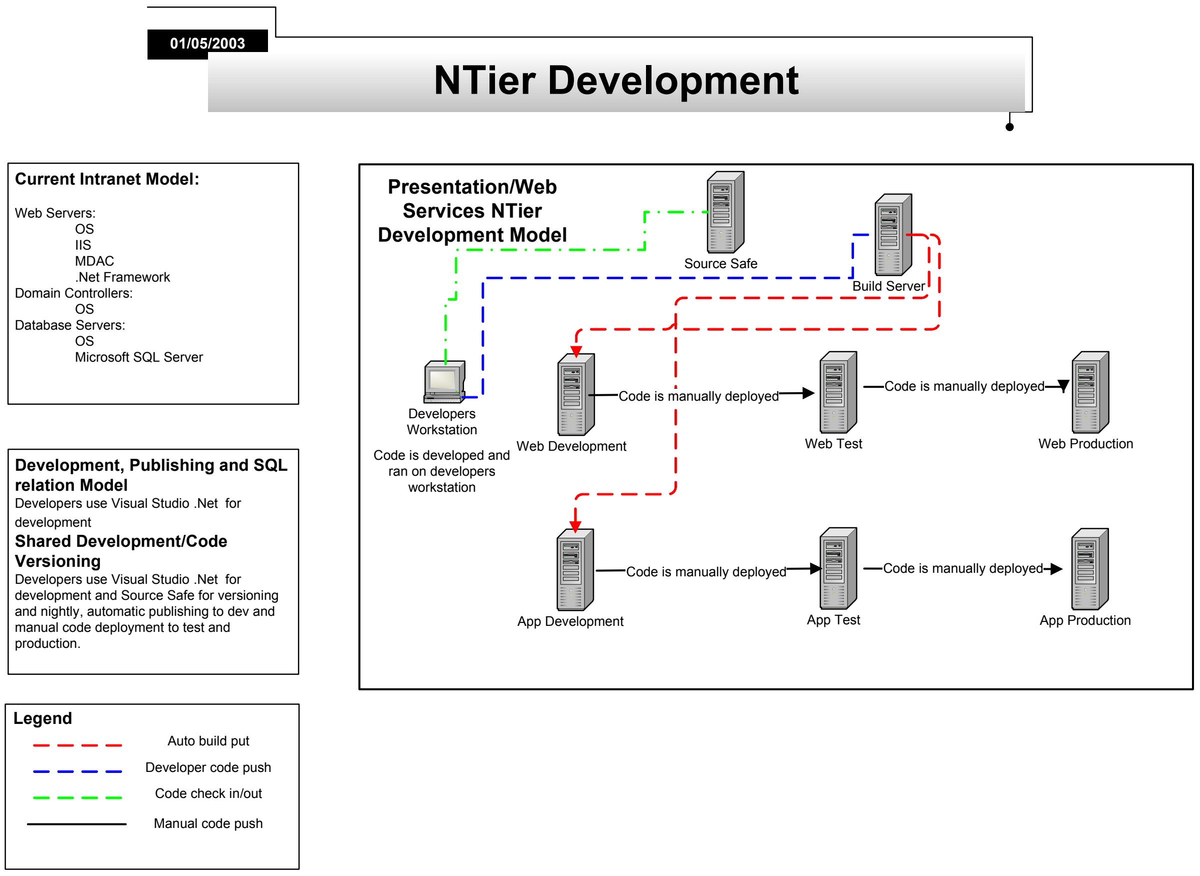
Physical Flow

The initial deployment of code will occur between the developers workstation and development servers. Subsequently, code will be deployed from test to production. The following diagram demonstrates the actual development model in the CDHS application environment.

`

Developer

⇒ **Figure 31 CDHS Physical Team Development**


01/05/2003
NTier Development
Presentation/Web Services NTier Development Model
Source Safe
Build Server
Developers Workstation
Code is developed and ran on developers workstation
Web Development
Code is manually deployed
Web Test
Code is manually deployed
Web Production
App Development
Code is manually deployed
App Test
Code is manually deployed
App Production

Current Intranet Model:

Web Servers:
- OS
- IIS
- MDAC
- .Net Framework

Domain Controllers:
- OS

Database Servers:
- OS
- Microsoft SQL Server

Development, Publishing and SQL relation Model
Developers use Visual Studio .Net for development

Shared Development/Code Versioning
Developers use Visual Studio .Net for development and Source Safe for versioning and nightly, automatic publishing to dev and manual code deployment to test and production.

Legend

🟥 – – –	Auto build put
🟦 – – –	Developer code push
🟩 – – –	Code check in/out
———	Manual code push

Movement of Code from Developer Workstation to Development Servers

The code deployment process is initiated by the developer using the following procedure:

First-Time Deployment

1. The developers opens a help desk ticket with the DHS Information Technology Help Desk and requests that the ticket be routed to the IU.
2. The IU will interview the developer to obtain details necessary to deploy code.
3. The IU will conduct an initial code review.
4. If there is an issue found in the code review, the developer will be notified and corrections must be made and the process re-initiated.
5. If the code passes the review, the code will be deployed to development servers.

Subsequent Deployments

1. Developers perform an xcopy deployment with no intervention by IU.

Movement of Code from Development to Test

First-Time Deployment

1. The developers opens a help desk ticket with the DHS Information Technology Help Desk and requests that the ticket be routed to the IU.
2. The IU will conduct an initial code review.
3. If there is an issue found in the code review, the developer will be notified and corrections must be made and the process re-initiated.
4. If the code passes the review, the code will be deployed to development servers.

Subsequent Deployments

1. The developers opens a help desk ticket with the DHS Information Technology Help Desk and requests that the ticket be routed to the IU.
2. The code will be deployed from development servers to test servers by the IU.

Movement of Code to Production

As the application moves through the development lifecycle the final step is the process of migrating the code from test to production. This step takes close coordination of the developers, the SSU and the IU. The Application Change Control process will be used to coordinate this final migration step

Procedures for Team Development

The following procedures are more detailed with respect to creating the work environment and requesting the build and promotion of code.

Visual SourceSafe Setup

The VSS server will be managed and maintained by the IU. The responsibility of creating the initial root folder for all the applications being hosted or developed in the CDHS environment will be that of the IU.

VSS on the Workstation

The folder structure and the naming of the folders would have already been established at this point in the application development process within CDHS. The creation of the application in Visual Studio should take place after the initial meeting with the IU and SSU. One of the main deliverables of this meeting is folder structure and the naming conventions for the application.

To create a solution and project, conduct the following steps using the agreed upon names from the meeting:

The Project Leader creates a new project including all needed sub-folders (e.g. for DATA, UTILITIES, BUSINESS etc.) as follows:

- Select menu Tools / Options / Projects / web Settings
    - Check - “File Share”
- Select menu Tools / Options / Source Control / General
    - Uncheck – check in everything when closing a solution
    - Uncheck – keep items checked out when checking in
    - Uncheck – allow checked in items to be edited
    - Select - Use: “Visual SourceSafe” settings in the combobox

VSS and Workstation Folder Structure

One of the most important factors when setting up a new application using the team development environment is the initial project structure. CDHS requires a consistent structure across on development workstations and the build server. To keep things simplified and symmetrical, a folder structure on the local developer workstation must match that of the structure on the Visual SourceSafe (VSS) Server.

By default, when creating a new ASP.NET Web application, the project file is located in a nominated virtual root beneath your default Web site (usually **\inetpub\wwwroot**) and the associated solution file is located beneath **My Documents\Visual Studio** Projects. This default arrangement is not ideal in a team development environment because it breaks the symmetrical structure between your VSS projects and local files and will not be supported at CDHS.

Standard Folder Structure

The ITSD standard is to develop solutions on the D: drive and map a virtual folder to your project in IIS.

- **D:\inetpub\applications\**<GroupName>\<SolutionName>**Solution**
- **D:\inetpub\applications\**<GroupName>\<SolutionName>**Solution \Project1**
- **D:\inetpub\applications\**<GroupName>\<SolutionName>**Solution \Project2**
- Etc.

Where <GroupName> is the name of your group
Where <SolutionName> is the name of your solution

Initial Project Creation of ITSD SourceSafe Solution

1. **IU** creates the initial root project folder for compiling the complete project
    a. Name folder as: <ProjectName>Solution.root, where <ProjectName> is the name of the project
        i. We add the "Solution.root" to the name as Visual Studio expects this when adding a project to source control

2. **LEAD DEVELOPER** creates the project on their local hard drive
    a. Create a new folder in: **D:\inetpub\applications\**<GroupName>\<SolutionName>**Solution**
        i. <GroupName> - name of your group (e.g. IUProjects for the Internet Unit)
        ii. <SolutionName> - name of your solution
            1. Always append "Solution" to the end as: TestSolution

    b. Open Visual Studio and create a new BLANK SOLUTION
        i. Save it under the directory created above

    c. In Visual Studio add all other projects to this solution as needed
        i. Create a sub-folder under the above solution folder using the project name
        ii. In IIS make this folder a virtual root
        iii. In Visual Studio add a new web project with the location of this folder: D:\inetpub\applications\<GroupName>\<ProjectName> Solution\<Projectname>

    d. Add Solution to SourceSafe
        i. In Visual Studio right-click on the solution file in the solution explorer and select "Add Solution to Source Control"

   ii. Select the Visual SourceSafe folder created for you by ITSD
      1. e.g. <ProjectName>Solution.root

3. **DEVELOPERS** may now open Visual Studio on their workstation and select to "Open project from Source Control" from the File / Source Control menu.

   a. **On Going Maintenance**
      i. Add References to any non-.NET core assemblies to Source Safe. If your project uses any assemblies (e.g. interop etc.) they must also be added to the bin folders in SourceSafe so that the build will succeed. (add to the \bin folder)

         1. Open SourceSafe and create a **bin** folder under the project where required

         2. Add each non .NET core DLL referenced in the project

      ii. Add to SourceSafe any new projects or files as development progresses. If needed also update any shared folders also by sharing the new items there.
4. **LEAD DEVELOPER** requests IU create a Build to be released to Development / Test / Production as needed using the defined procedures.

Other SourceSafe Project Structures

The above structure should be suitable for most small to medium size projects where each developer has the whole solution created on their local hard disk and in Visual Studio. For larger projects where the team leader would like to isolate developers, contact the Internet Unit.

Code Check-In/Out

Understanding the procedures for code check-in and check-out are important in a team development environment. The following identifies the procedure that is to be followed in CDHS.

Code Check-In

- Only check in code that complies without errors (warnings are ok),
- Always Check-In at the end of the day. On Check-IN always LABEL the project using the current assembly version. This will make it easy to roll back to a previous build.

Note: If you cannot compile the project without errors at the end of the day, at least back it up onto a network share BUT DO NOT check it in

Check-Out

- Always get latest version of all projects (solution) at start of day,
- Check-out your own projects only as needed,
- Only check out the solution file for short periods to add or remove projects.

Assemblies

Referencing Assemblies

1. Make references by project rather than by File if possible. The advantages of using project references are:

   - They work on all development workstations where the solution and project set are loaded. This is because a project Globally Unique Identifier (GUID) is placed in the project file, which uniquely identifies the referenced project in the context of the current solution.
   - They enable the Visual Studio .NET build system to track project dependencies and determine the correct project build orders.
   - They avoid the potential for referenced assemblies to be missing on a particular computer.
   - They automatically track project configuration changes. For example, when you build using a debug configuration, any project references refer to debug assemblies generated by the referenced projects, while they refer to release assemblies in a release configuration. This means that you can automatically switch from debug to release builds across projects without having to reset references.
   - They enable Visual Studio .NET to detect and prevent circular dependencies.

2. Use file references only when necessary. If you cannot use a project reference because you need to reference an assembly outside of your current solutions project set, you must set a file reference.

Include Outer System Assemblies with Projects

The best way to handle outer system assemblies, such as third-party Web controls or components that are not rebuilt by your build process if approved, is to include them directly in those projects that need to reference them. Conceptually, think of outer system assemblies the same way as .bmp or .gif files.

To include and then reference an outer-system assembly, perform the following:

1. In Solution Explorer, right-click the project that needs to reference the assembly, and then click Add Existing Item.
2. Browse to the assembly, and then click OK. The assembly is then copied into the project folder and automatically added to VSS (assuming the project is already under source control).
3. Use the Browse button in the Add Reference dialog box to set a file reference to assembly in the project folder.

The advantages associated with using this approach are:

1. The file can be added to VSS as a new file version complete with its own file history if outer system assemblies remain source controlled alongside the project files
2. The entire system is contained within VSS; including all outer-system assemblies, such as third party controls. You can retrieve an earlier version of the system from VSS, including all source code and external dependencies. This allows developers to have a complete snapshot of the earlier system version.

Sharing Assemblies

In order to avoid difficulties in updating an assembly to a later version when a particular outer system assembly is referenced by multiple projects, share the outer system assembly is VSS between the projects that use it. This allows the file to be updated within one project. The developer can refresh the copies maintained within other projects by right-clicking the project within Solution Explorer, and then clicking Get Latest Version (Recursive).

SourceSafe Security Issues

Depending on your SourceSafe structure you may wish to further secure the SourceSafe folders beyond the default used by ITSD. This can be done on a folder-by-folder or project-by-project basis. In no case can it be done on a file-by-file basis.

VOLUME VI: CHANGE CONTROL

This volume addresses the change control process that supports the application environment at CDHS. It covers both infrastructure and application change control processes.

The fundamental purpose of change management is to establish and maintain the integrity and control of software and hardware products throughout a project's life cycle. The CDHS ITSD has developed both an infrastructure and application change management process.

The key integrated aspects of the CDHS change management processes are:

- Management of the change process to ensure product and environment integrity
- Identification of the baseline
- Status accounting
- Audit trail

It is important to understand that the scope of the change processes as identified by ITSD is to protect the integrity of the hosting environment. It is essential for application customers to ensure that sufficient change management, quality assurance, and testing processes are in place to protect the integrity of the application and its data as it relates to the business function the application supports.

[4]Section I Application Change Management

There are a number of web-based applications hosted within the California Department of Health Services (CDHS) Information Technology Services Division (ITSD) environment. A standard change management (CM) process is vital to the stability and health of the overall environment and its users because the volume and complexity of the applications hosted.

This process provides a method for complying with the provisions of the Health Administrative Manual, Section 61040.7, State Administrative Manual, Sections 4946, and Statewide Information Management Manual Configuration Management, Change Control.

The application environment serves development, test and production. The main purpose of this Change Management procedure is to protect the integrity of the production hosting environment. Therefore, any change being made to this environment must follow the steps detailed in this document.

Change Management Objectives

The purpose of change management is to execute changes in a rational and predictable manner so that staff and clients can plan accordingly. The extent of a proposed change requires equivalent effort in forethought, monitoring, and post-implementation evaluation. This enables project owners and customers to avoid negative consequences while increasing the value of information resources.

The Information Technology Services Division (ITSD) hosts custom and commercial off-the-shelf (COTS) applications. All of these utilize shared infrastructure resources. The sharing scope may include network, server, software and support staff. The impact of a particular application or application component can have an unforeseen impact on the environment. Implementing a change control process enables ITSD to document, track and respond to events that may impact core resources. This Application Change Management process provides the following:

1. Ability to maintain and improve system reliability, availability, serviceability and functionality.
2. A method for users of the hosting services to submit a request for change including introducing, modifying or retiring a system.
3. Identification of the entity requesting the change and which application is subject to the change.

[4] This section of this document will later be extracted. However, since this is the first time this process s being introduced it is being incorporated in its entirety.

4. A method for defining the change event plan.
5. A documented recovery or rollback procedure in the event that the change is not successful.
6. An opportunity to assess the risk that the change may introduce to the system or environment.
7. A communication process amongst those affected by the proposed change.
8. Identification of scheduling conflicts for changes.
9. A historical record of changes occurring in the hosting environment.
10. Enhancement of management and technical staff awareness of the above.

In-Scope Items

The scope of this application change process includes, but is not necessarily limited to the following changes. In the event that there is any doubt, a change request should be submitted:

1. COTS and Custom Application:
    a. Introduction of New
    b. Modification of Existing
    c. Retirement or Transition Hosting of Existing

2. Environment Configuration:
    a. Application Security Modifications
    b. Underlying Support Product Configuration Changes
    c. Creation or Deployment of Third-Party Tools

3. Database:
    a. Database Modifications
    b. Database Object Creation

Roles and Responsibilities

The Change Management members are intended to represent the groups that are responsible for web, application and database layers for web based applications, custom or commercial, hosted by the ITSD.

The Change Management members and their responsibility are as follows:

Role	**Qualification**	**Responsibility**
CM Lead *Change Management Lead*	An individual identified as a Product Manager, Project Manager, Senior Architect or Developer for an application.[5]	· Prepares and submits accurate change submission request. · Ensures that all appropriate entities identified in their project structure approve of the change and commitment of resources to do so obtaining signatures for the requested

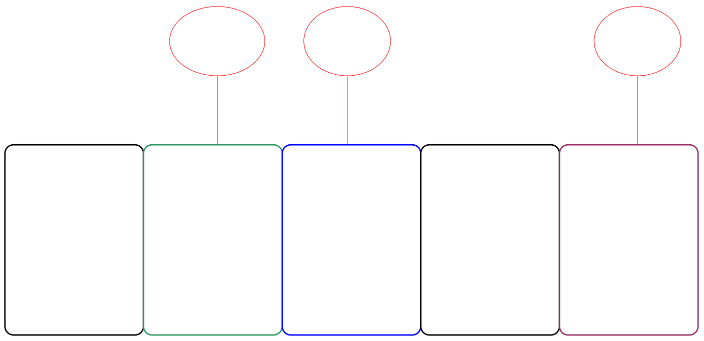
[5] A Product Manager or Project Manager should have a sufficiently qualified technical person present when discussing the change details.

Role	Qualification	Responsibility
		change if necessary. · Sends Calendar notices to all technical parties involved for the change window. · Manages the communication channel for the execution of steps in completing the change one officially approved and during the scheduled time. · Manages communication with the end-user community.
CM Coordinator *Change Management Coordinator*	ITSD appointed Senior or Staff level technical representative.	· Conducts initial review of change request, providing feedback to CM Lead. · Makes initial change factor level assignment and assigns change number. · Acts as the manager for the change documents and schedule. · Adds the request to the schedule for change activity. · Facilitates the change meetings.
CM Committee *Change Management Committee*	· Senior or Staff level technical representatives from ITSD Internet Unit, SQL Server Unit, and Information Security Office. · Network, Desktop, Messaging, or other representative (s) will attend as needed.	· Reviews the requests for test, implementation, documentation, scheduling and recovery implications. · Assesses request for environment impact. · Assesses reasonableness of the initial change level factor. · Assigns appropriate technicians for changes. · Negotiates schedule.

Change Management Timeline

The Change Management group will formally meet every Tuesday at 10:30 a.m. to review change requests at the Change Management Review Meeting (CMRM). Change Requests for the Tuesday meeting are to be submitted to the CMC by close of business the previous Wednesday. The CM Coordinator is to give the requestor feedback no later than close of business Friday on the initial **Assessment** as described in subsequent sections.

⇒ Figure 32 Change Management Timeline



Required Working Documents

The CM Committee will determine the appropriate action to be taken for each change request. To make the informed decisions the CM Lead is to prepare and adjust the following documentation, with assistance from CM Committee members if necessary, for the CM Committee:

1. Change Request
2. List of Tasks to be Performed
3. Schedule
4. Change Status Form
5. Additional Support Documentation as Required by the CM Committee

Change Factors

The assessment of change factors is essential to the processing of a change request. The CM Lead has the most knowledge about the application level change as it relates to the business function of their represented organization while the CM Coordinator has the most knowledge about the change as it relates to the aggregate of environment changes. Together, the two parties must contribute to the assessment of change factors. The following define the change factors that are considered:

Mon

1. Risk
2. Environment Impact
3. Communication Requirements
4. Installation Time
5. Documentation Requirements
6. Education/Training Needs

<u>Risk</u> for the purposes of this process is the probability that the proposed change will have a negative effect on the production environment, probability of success on first attempt, and the complexity of recovery in the event of failure.

Environment Impact is the identification of events that may be triggered as a result of a change to the hosted application.

Communication Requirements takes into account the number of potential contacts that must be notified of the change as they may be affected as a result of the proposed change.

Installation Time addresses the total effort to prepare for the change, implement it, test its success, recover from failure if needed and notify affected parties.

Documentation Requirements it is the assembly of information about the change and recovery from it if necessary with the intention of the CM Committee to retain accurate documentation about applications that ITSD hosts as prepared by the CM Committee and the CM Lead.

Education/Training Needs considers how significant the education or training efforts will be for the support staff responsible for supporting the application.

Emergency Exception is a way in which vital changes that need to happen very quickly can be accomplished. It is the responsibility of the CM Lead to alert the CM Coordinator that the request is an emergency. It is the CM Coordinator's responsibility to obtain ITSD Infrastructure Support Branch management approval understanding that this management team will coordinate with higher levels of management if necessary. All Emergency changes require full documentation, but because of the reduced timeline the documentation may occur after the fact.

Change Factor Ratings

As described in the subsequent process, the criteria above will be evaluated to develop a Change Level Factor Rating. The ratings are: Level 1, Level 2 Level 3 or Level 4 as defined by the following:

Level 1 is **highly complex in nature**, requiring a minimum of two weeks advanced notice, based on the following:

1. Potential impact to system availability to large portion of the hosted or user community
2. May be disruptive to the production environment
3. Complex notification base
4. Complex installation or recovery process
5. A large number of support staff need to be involved with significant coordination required
6. Comprehensive nature of documentation required to support the change
7. Lack of knowledge of support staff

Level 2 is of **medium complexity**, requiring standard notification based on the following:

1. Involves general application or database changes that are more typical in nature including new application releases, database modifications, tool upgrades and general configuration changes.
2. Is low risk for production disruption
3. Requires little notification
4. Has a routine-in-nature installation and recovery process
5. Requires little coordination
6. Requires little documentation
7. Staff are familiar with the technologies and support of them

Level 3 is of **low complexity** requiring standard notification based on the following:

1. Changes can be accomplished quickly with no perceived impact or risk
2. Normally non-disruptive or administrative in nature
3. Require a small number of individuals to be involved
4. The impact of failure is either highly unlikely or minimally limited in scope
5. Requires limited coordination
6. Requires little documentation
7. Staff are familiar with the technologies and support of them

Level 4 is executed as a result of an operational emergency as deemed by the CM Lead with feedback from CM Committee members when feasible and may be executed in an expedited manner.

Tracking Requests

It is the intention of the CM Committee to provide an easy method of open communication. This includes the ability for CM Lead to easily track the status of changes they submitted. This is accomplished by having a master change list in which all approved change requests are posted. This allows both the CM Lead and the CM Committee to have a central repository of activity and schedule. Each request is given a unique number for tracking the change throughout the process.

Change Management Process

The change management process is designed to allow hosted customers to request deployment, modification, or retirement of a custom or commercially purchased application. The process is intended to allow the customer to communicate procedures for implementation of the change. At the same time, the hosting entity assesses the impact on the environment and supporting resources to eliminate any issues enabling successful change or recovery in the event that a change does not result in the desired outcome. In addition, both parties are kept abreast of the progress and historical record of the changes.

The Change Management process consists of several phase consisting of the following:

1. Preparation and Submission
2. Assessment
3. Concept Authorization or Rejection
4. Planning
5. Change Authorization or Rejection Meeting
6. Scheduling
7. Implementation
8. Post Implementation Review

Preparation and Submission

The **Preparation and Submission** phase includes a CM Lead completion of a change request and submission to the CM Coordinator.

Assessment

In the Assessment phase, the CM Coordinator assigns a tracking number and reviews the request form for completeness. It is the intent of the CM Process to improve the efficiency with which changes are implemented for ITSD and its customers and requires structure.

Complete

If complete, the CM Coordinator makes an initial Change Level Factor classification. The Change Request is made available to the CM Committee for review and is placed on the Change Items list for discussion at the Change Control meeting.

Incomplete

If incomplete, the request preparer will be notified that modifications are necessary.

Concept Authorization or Rejection

The Concept Authorization or Rejection phase solidifies the approval or denial of a proposed change as a concept. Approval solidifies the intent of all parties to dedicate resources to the review effort to assess the impact and risks associated with the proposed change. Rejection solidifies the intent of all parties to cease the execution until sufficient amendments are made or altogether.

Planning

The Planning phase deliverables include identification of specific tasks, event sequences and responsibilities for completion of a successful change. It serves to minimize oversight of tasks, dependencies or scheduling issues.

Change Authorization or Rejection Meeting

The Change Authorization or Rejection Meeting phase solidifies the approval or denial of a proposed change. Approval solidifies the intent of all parties to execute the steps defined in the request with an accurate view of the risks and impact that the change may have on the environment. Rejection solidifies the intent of all parties to cease the execution altogether or until sufficient amendments are made.

During this meeting the group requesting the change will give an overview of the change request. Immediately following questions can be presented and concerns voiced. The CM Committee will approve or reject the change request.

If the request is approved the official schedule can be proposed by the requestor. Once final approval has been made the change is added to the master change schedule with a set date for the change to be made.

If the change is rejected, the request is returned to the author for further action as identified in the meeting. Examples include, amending proposed schedules, providing additional information in support of the change, and modification of recovery procedures.

Scheduling

The Scheduling phase deliverables ensure that all parties essential to completion of the changes are authorized to perform the work and have allocated the appropriate time and resource to execution of the change. All changes are to be scheduled and performed based on the master activity schedule. This allows resource conflicts to be identified and alternate arrangements to be made when necessary.

Implementation

The Implementation process establishes a mechanism for which changes are applied and reversed if critical success checkpoints have failed.

It is the CM Lead's responsibility to ensure that their user community is aware of any proposed changes in advance, providing them with instructions on how to notify the appropriate Program contact to report issues. In the event that the change is driven by infrastructure needs, the responsibility lies on ITSD for notification to Program contacts.

Post–Implementation Review

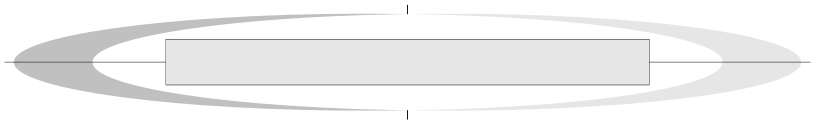
The Post-Implementation Review process is a mechanism for the CM Committee to evaluate if the change occurred in accordance with the process. Process amendments may result from this. This process ensures that:

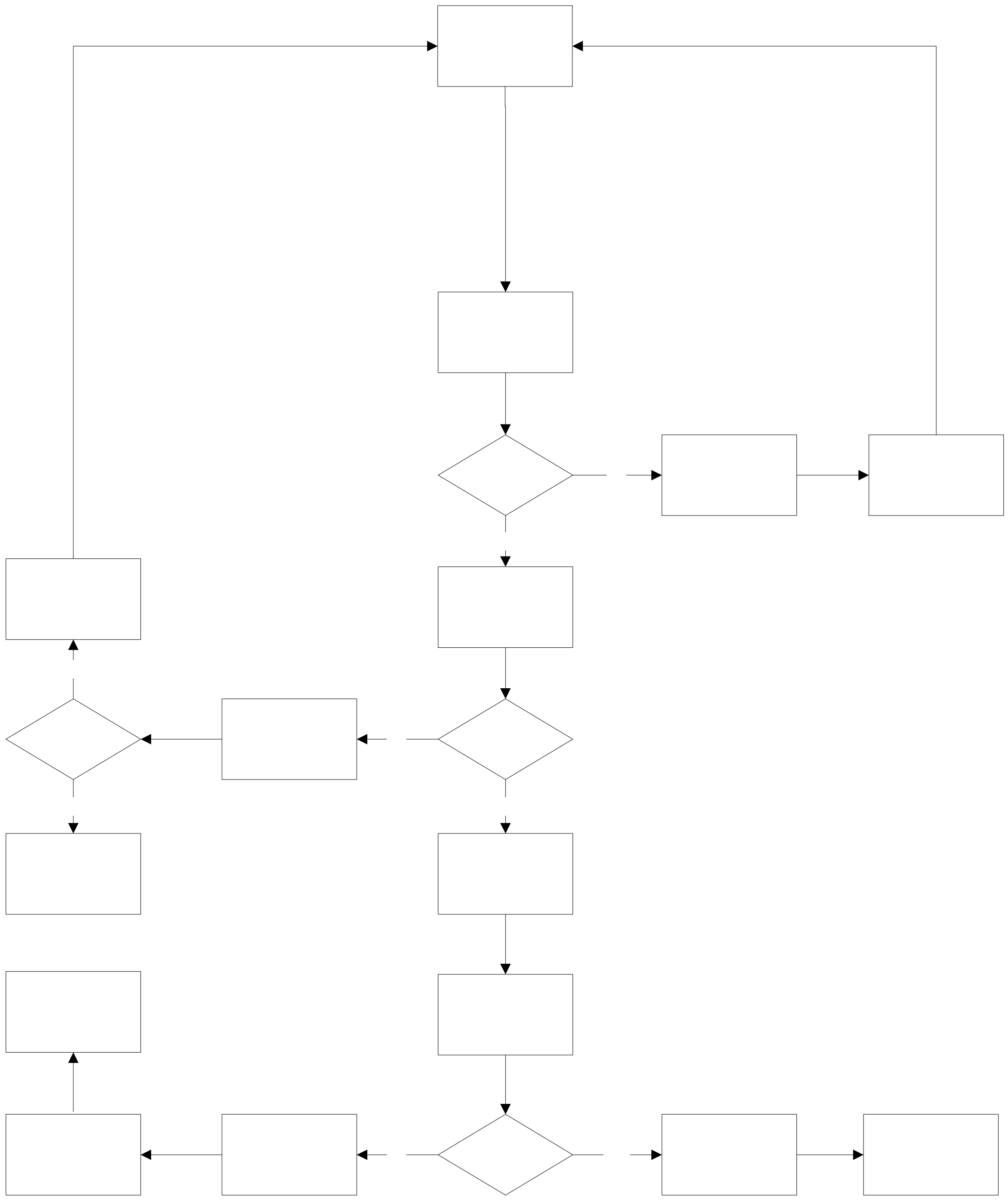
1. Change procedure was followed
2. Processes adequately enabled fulfillment of the objectives or recovery if necessary. If not, an opportunity to modify the process is present.
3. Changes are assigned a final and documented status of complete, failed, backed out or cancelled.

Application Change Management Processes Diagrams

The following diagrams show the process for executing a change request.

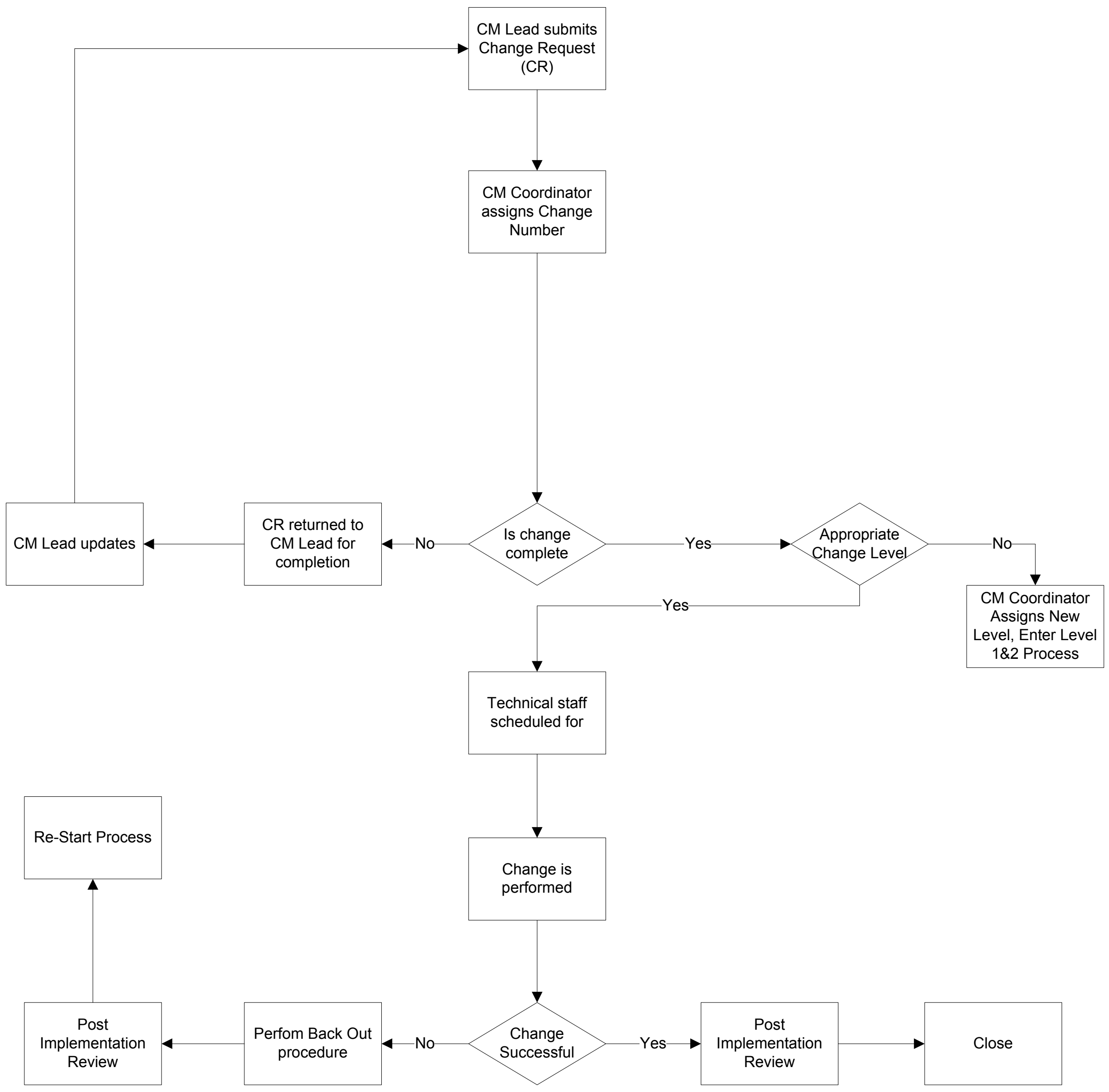
Level 1 & 2, High and Medium Complexity



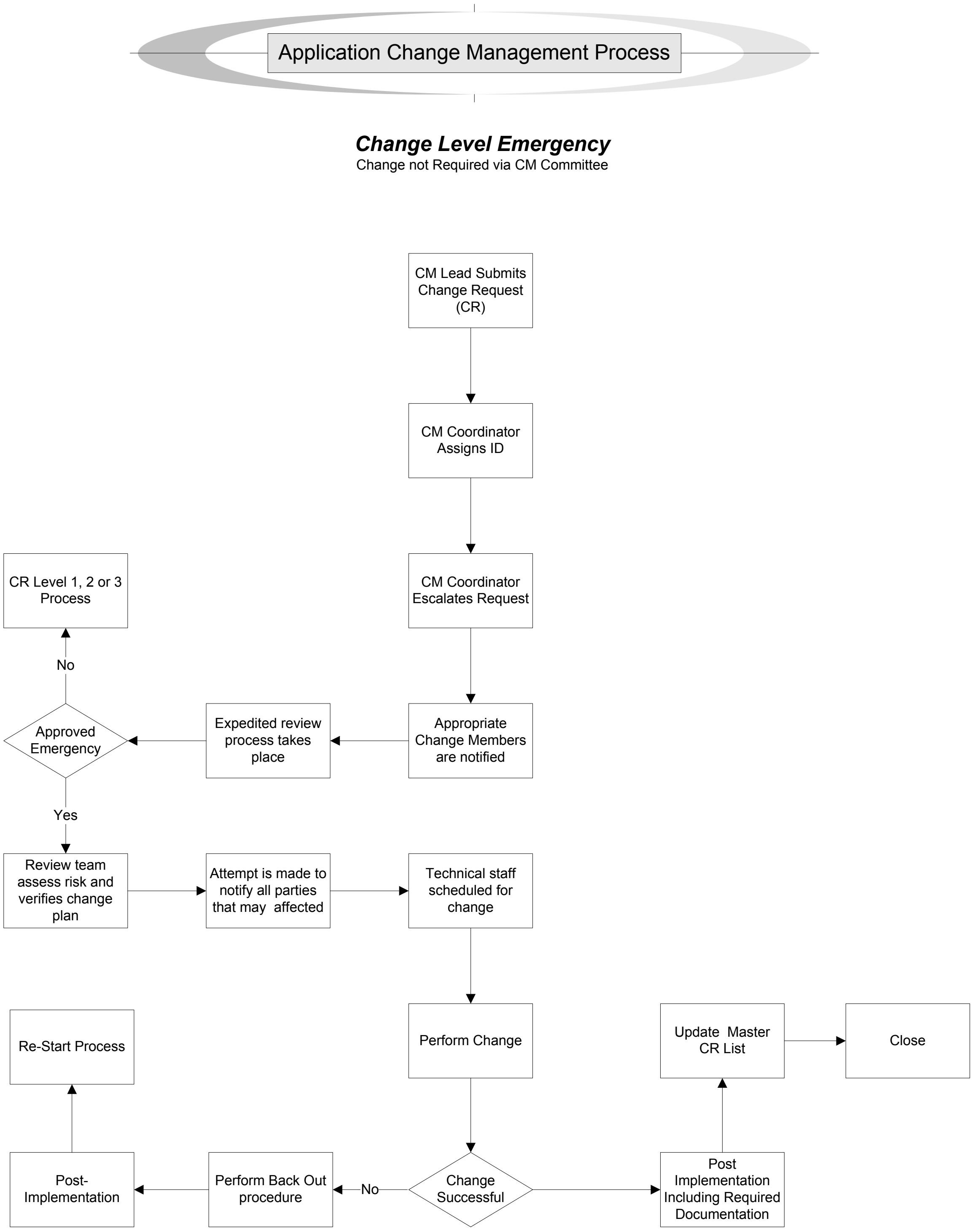


Level 3, Low Complexity

Change Level 3
Change not Required via CM Committee


CM Lead submits Change Request (CR)
CM Coordinator assigns Change Number
Is change complete
No
CR returned to CM Lead for completion
CM Lead updates
Yes
Appropriate Change Level
No
CM Coordinator Assigns New Level, Enter Level 1&2 Process
Yes
Technical staff scheduled for
Change is performed
Change Successful
No
Perfom Back Out procedure
Post Implementation Review
Re-Start Process
Yes
Post Implementation Review
Close

Level 4, Emergency


Application Change Management Process
Change Level Emergency
Change not Required via CM Committee
CM Lead Submits Change Request (CR)
CM Coordinator Assigns ID
CM Coordinator Escalates Request
CR Level 1, 2 or 3 Process
No
Approved Emergency
Expedited review process takes place
Appropriate Change Members are notified
Yes
Review team assess risk and verifies change plan
Attempt is made to notify all parties that may affected
Technical staff scheduled for change
Re-Start Process
Perform Change
Update Master CR List
Close
Post-Implementation
Perform Back Out procedure
No
Change Successful
Post Implementation Including Required Documentation

Standard Application Change Procedure

The following summarizes a typical change process as it is implemented.

The basic process for each change is as follows; however, additional details are often necessary and should be addressed using the Additional Procedures section that follows:

Standard Change and Recovery Procedure	**Standard Change and Recovery Procedure Detail**
Perform Baseline	❖ Customer is to validate working baseline of the application prior to change. ❖ IU staff will perform baseline statistics on server performance if required. ❖ DBA will perform baseline statistics of database performance if required.
Perform Complete Backup	❖ The IU staff person will copy all user interface and business logic components to a back up directory on a local drive of the web and application servers if required. ❖ During this process the DBA from the SSU will perform a complete back up and store it on the local drive of the server if required.
Perform Upgrade	❖ The IU staff person will perform the approved change request if required. ❖ The DBA will perform the approved change request if required.
Infrastructure Test of Application	❖ The IU staff person will test basic connectivity. ❖ The DBA will conduct test basic functions: o Verify database is online o Use Query Analyzer to connect to the database o Run simple query to verify connectivity
Infrastructure Assessment	❖ IU and DBA staff will determine if the change requires rollback due to any infrastructure related issue. If no issue detected, Test Application Integrity step occurs. If a significant issue is noted, Rollback step occurs.
Test Application Integrity	❖ The change requestor or designated test staff will log in via the application to verify connectivity from the application to the database using a test script developed

Standard Change and Recovery Procedure	**Standard Change and Recovery Procedure Detail**
	by the customer in advance to validate success of changes. ❖ Change requestor will notify IU and SSU staff if any problem arises during testing. ❖ Change requestor will determine if change is successful or requires rollback.
Rollback	❖ The web user interface, business logic, or web/application server configuration changes are restored to the initial state using pre-change documentation and backed-up files. ❖ The database is immediately restored from the complete back up. ❖ The change is terminated and Post-Implementation Change Status Details form is completed.

Application Change Request Form

Please complete all of the following areas unless noted as CC use only. This is a two part form.

Part A – Change Summary

Change Requestor		**Requestor's Phone**	
Requestor's Email		**Application Name**	
Proposed Change Level	☐1 ☐2 ☐3 ☐4	**Date Requested**	
Project Billing Code		**Division, Branch, Unit**	
Estimated Impact	☐H ☐M ☐L ☐Unk	**Estimated Completion**	
Change Type			
Staff Needed	☐Internet[6] ☐Database ☐Network ☐Server ☐Help Desk ☐ Messaging ☐Client ☐Project Office ☐Other, Specify		
Purpose of Change			
Description of Change			
Requestor Notes			

Supporting Documentation	*Required* ☐Change Recovery Plan ☐Detailed Change Instructions	*Other, Please Specify*

For CC use only

Change Number			
Change Approved	☐Y ☐ N ☐ P ☐D	**Change Level Assigned**	☐1 ☐2 ☐3 ☐4
Change Date		**CC Coordinator**	
Change Status	☐Successful ☐Successful with unexpected issues	☐Postponed/Cancelled ☐Unsuccessful	
Change Notes			

[6] Internet staff is associated with staff that support the user interface and business logic servers for the application.

Part B – Change Detail

Database Change Detail

The following details the changes that are being made to the production database for the application.

Change Risk	☐H ☐M ☐L ☐Unk	**Estimated Time to Complete Change**	
Risk Assignment Reasoning			
Reason for Change			
Testing Method			
Detailed Description of Change			
Resources Needed	*Internet Staff*	*SQL Server Unit*	*Other Staff*

For CC Use Only

Change Reviewed By		**Changed Approved By**	

Application Change Detail

The following details the changes that are being made to one or more of the following:

- Application User Interface
- Application Business Logic
- Commercial Web Product

Risk of Change	☐H ☐M ☐L ☐Unk	**Estimated Time to Complete Change**	
Risk Assignment Reasoning			
Reason for Change			
Testing Method			
Type of Application	☐Custom ☐COTS	**Change Interface**	☐User Interface ☐Web Service ☐Business Logic
Detailed Description of Change			
Resources Needed	*Internet Staff*	*SQL Server Unit*	*Other Staff*

For CC Use Only

Change Reviewed By		**Changed Approved By**	

Change Status Supplemental Forms

The following supplemental forms may be required as part of a completed change request package.

Change Recovery Plan Form

The following information describes the recovery plan for both the database and application in the event that the standard procedures identified in Appendix B are not sufficient to recover from a change that is not successful.

Additional Procedures

Additional Step(s)	
Details	
Reason(s)	

Detailed Change Instructions Form

Please list all steps that were taken by the developers to perform the requested change. In addition to the steps taken to perform the change please list out the instructions for deploying the proposed change. Fill out all sections that apply.

Database	
List all DB objects added or modified	
Special Instructions List any non standard instructions for deployment of objects listed above	

Application	
List all new application changes	
Special Instructions List any non standard instructions for deployment of objects listed above	

Post-Implementation Change Status Details Form

In the event of a proposed change being assigned a change status of unsuccessful the following form must be completed before a new Change Request can be rescheduled.

Change Number		**Change Date**	
New proposed change date if any			
List all individuals involved in change			
Reason for unsuccessful change			
Did it cause production down time	☐Y ☐ N ☐Unk	**Did Roll Back Plan succeed**	☐Y ☐N ☐Unk
Was all production data retrieved	☐Y ☐ N ☐Unk	**Was the testing plan adequate**	☐Y ☐ N ☐Unk
Future steps that can be taken for successful change			

Infrastructure Change Control

ITSD has implemented a Infrastructure Change Control Committee to formally manage changes to the CDHS production environment. It is used for review and approval of all requests for change. Participation by all levels of ITSD technical staff will help to ensure that the division provides the best possible standards for information technology services. The Infrastructure change control information is documented, maintained and available in the CDHS Exchange Public Folders located at: outlook:\\DHS Public Folders\DHS-Programs\ITSD\ITSD - Public\ITSD-Change Control .

VOLUME VII: APPENDICIES

The following appendices contain essential information regarding additional services or supporting background information in utilizing the CDHS environment for hosting services.

Appendix A Web and Database Hosting Questionnaire

This document contains a four-part questionnaire to be completed during the Design phase of your project. The components of the questionnaire are:

1. Project Information
2. Application Details
3. Technical Diagrams, and
4. Systems Documentation

If you have any questions about completing this form, please contact the following groups via the CDHS Information Technology Help Desk at: http://remedyhelp.dhs.ca.gov:

- For Database related questions including data and data access layers: SQL Server Unit
- For Web Hosting related questions including presentation and business logic layers: Internet Unit

Project Information

Please provide the following information for the purpose of assisting us in coordinating project management activities:

Project Identification Information

Project Name: **Project CAB Code:**

Project Contact Information

Project Sponsor

Name:	
Title:	
E-mail Address:	
Office Phone:	
Mobile/Pager:	

Project Manager

Name:	
Title:	
E-mail Address:	
Office Phone:	
Mobile/Pager:	

Program Technical Lead

Name:	
Title:	
E-mail Address:	
Office Phone:	
Mobile/Pager:	

ITSD Project Management Office Representative

Name:	
Title:	
E-mail Address:	
Office Phone:	
Mobile/Pager:	

ITSD Application Support Branch Project Manager (if applicable)

Name:	
Title:	
E-mail Address:	
Office Phone:	
Mobile/Pager:	

Application Details

The following section is designed to capture details related to the application hosting requirements.

ID	**General Information**
1.	Will this application be considered "mission critical"? (**Please check one)** ☐ Yes ☐ No
2.	Is this application? **(Please check one)** ☐ Web Interface based ☐ Windows Interface based ☐ Other
3.	Application provides access to information that is: **(Please check one)** ☐ Confidential ☐ Sensitive ☐ Private ☐ Public
4.	Anticipated application lifecycle duration? **(Please check one)** ☐ Single event ☐ Annual event for specified duration ☐ Continuous business function
5.	Availability expectation: **(Please check one)** ☐ 24/7 ☐ Business Hours Only (M-F, 7:00 a.m. – 5:00 p.m.) ☐ Other, please specify:

ID	General Information
6.	Application provides access for: **(Please check one)** ☐ 1-25 users ☐ 26-75 users ☐ 76-150 users ☐ 151+ users, please specify:
7.	Application provides access to: **(Please check all that apply)** ☐ Public ☐ CDHS Employees ☐All employees ☐Workgroup(s) only ☐ Other State Agencies, please specify: ☐ Business Partners, please specify:
8.	Application provides access to end-users from: **(Please check all that apply)** ☐ Internal CDHS network ☐ East End Complex ☐ Richmond Lab Complex / Bay Area ☐ Southern California ☐ Other, please specify: ☐ Remote Access through VPN ☐ Employees ☐ Consultants Working in Employee Capacity ☐ Non-Employee/Consultants Working in Employee Capacity Users
9.	Were you given a copy of the Information Technology Architecture Standards? ☐ Yes, Version Date: ☐ No

ID	**Application Hosting Details**
1.	Where will this application be hosted? **(Please all that apply)** ☐ ITSD ☐ Lab ☐ Development ☐ Test ☐ Production ☐ HHSDC ☐ Lab ☐ Development ☐ Test ☐ Production ☐ Other, specify: If you will not be using ITSD managed enterprise resources for development, test and production purposes, who will manage the following: Infrastructure Servers Application Components/Services Web Components/Services Database Other:
2.	The proposed uniform resource locator (URL) is:
3.	Is this web application or site related to a media campaign? ☐ Yes, Office of Public Affairs contact is: ☐ No
4.	User interface screens estimated: **(Please check one)** ☐ 10 or less ☐ 11- 50 ☐ 51-100 ☐ 100+, please specify:

ID	Application Hosting Details
5.	Application data transactions per year: **(Please check one)** ☐ 10 thousand or less transactions ☐ 11 - 25 thousand transactions ☐ 26 - 50 thousand transactions ☐ 51 - 100 thousand transactions ☐ 100+ thousand transactions
6.	Will this application require database support outside normal business hours? **(Please check one)** ☐ Yes ☐ No
7.	Is MS Access part of this application? **(Please check one)** ☐ Yes ☐ No
8.	Is Business Intelligence part of this application? **(Please check one)** ☐ Yes ☐ No
9.	Will you require site use reporting services? **(Please check one)** ☐ Yes ☐ No
10.	Will CDHS own the source code in a non-compiled format? **(Please check one)** ☐ Yes ☐ No

ID	Application Development & Deployment Details
1.	Will the application use exception handling via? **(Please check one)** ☐ In global.asax ☐ On page
2.	Will the application use custom error pages? **(Please check one)** ☐ Static messages ☐ Showing runtime errors

3.	If displaying user input as output: **(Please check all that apply)**
	
	☐ Validating all user input (for script tags etc.)?
	    ☐ Yes
	    ☐ No
	☐ HTML-encoding output?
	    ☐ Yes
	    ☐ No

4.	Will the application contain any sensitive data in code? **(Please check one)** ☐ Yes ☐ No
5.	Will you require testing services? **(Please check all that apply)** ☐ Database Baseline ☐ Database Performance ☐ Database Server Performance ☐ Application Baseline ☐ Application Performance ☐ Application Server Performance ☐ Web Server Performance ☐ System Integration Testing ☐ Network Performance ☐ Acceptance Testing ☐ Other, specify:
6.	Identify the programming languages you intend to use: **(Please check one)** ☐ C# ☐ VB.NET ☐ Other, specify:

ID	**Application Development & Deployment Details**
7.	Will the application meet the minimum State requirements for accessibility as further defined by the World Wide Web Consortium (W3C) – http://www.w3.org/WAI? **(Please check one)** ☐ Yes ☐ No, explain:
8.	Will the application target browsers and versions other than those specified in the State of California standards for public access? **(Please check one)** ☐ Yes ☐ No, explain including versions of CSS, HTML, etc that you will use:
9.	What is the intended saving state? **(Please check all that apply)** ☐ App Object estimated size: ☐ Session , estimated size: ☐ Cache , estimated size: ☐ Viewstate, estimated size (largest page): Estimate # of concurrent users Normal : Estimate # of concurrent users Peak: Peak Frequency: If crafted for web farms, select where session will be saved: ☐State Service ☐SQL Server

ID	Systems Integration Details
1.	Does this Application integrate with any other systems? (for example: Exchange, Biztalk, SharePoint, Remedy, Mainframe, State Agency, etc.) **(Please check one)** ☐ Yes, please specify: ☐ No
2.	Does this application require FTP (File Transmission Protocol) services for file transfer? **(Please check one)** ☐ Yes ☐ No
3.	System supports ability to control import and export of data. **(Please check one)** ☐ Yes ☐ No ☐ N/A
4.	System provides web applications that require access-using protocols (such as HTTP, Telnet, NNTP, or FTP). **(Please check one)** ☐ Yes ☐ No ☐ N/A
5.	FTP, if allowed inbound, should not allow anonymous FTP. **(Please check one)** ☐ Yes ☐ No ☐ N/A ☐ If yes, specify:

ID	Additional Services
1.	Does system require the use of e-mail or online text messaging services? **(Please check one)** ☐ Yes ☐ No

ID	Database Security
1.	Is auditing a requirement or function within this application? **(Please check one)** ☐ Requirement ☐ Function ☐ No Auditing Requirement
2.	Does this Application audit user insert activity? **(Please check one)** ☐ Yes ☐ No
3.	Does this Application audit user update activity? **(Please check one)** ☐ Yes ☐ No
4.	Does this Application audit user delete activity? **(Please check one)** ☐ Yes ☐ No
5.	Does this Application audit logins? **(Please check one)** ☐ Yes ☐ No

ID	Database Functions
1.	Does this application use SQL "Systems / Extended Stored Procedures"? **(Please check one)** ☐ Yes ☐ No
2.	Does this application use SQLXML? **(Please check one)** ☐ Yes ☐ No
3.	Does this application use VIEWS? **(Please check one)** ☐ Yes ☐ No
4.	Does this application use SQL "User Defined Data Types"? **(Please check one)** ☐ Yes ☐ No
5.	Does this application use SQL "User Defined Functions"? **(Please check one)** ☐ Yes ☐ No
6.	Does this application use SQL "Roles"? **(Please check one)** ☐ Yes ☐ No
7.	Does this application use MS Analysis Services? **(Please check one)** ☐ Yes ☐ No
8.	Does this application use SQL "Full-Text Search"? **(Please check one)** ☐ Yes ☐ No
9.	Does this application use SQL "Rules"? **(Please check one)** ☐ Yes ☐ No

ID	**Database Functions**
10.	Does this application include any Data Definition Language (DDL) commands? **(Please check one)** ☐ Yes ☐ No
11.	Does the database schema contain referential integrity? **(Please check one)** ☐ Yes ☐ No
12.	Does the database schema contain column constraints? **(Please check one)** ☐ Yes ☐ No
13.	Does this Application require file-level and column-level data encryption? **(Please check one)** ☐ Yes ☐ No

ID	Database Jobs
1.	Are data imports and/or exports required for this application? **(Please check one)** ☐ Yes, please select all that apply: ☐ Scheduled ☐ On-Time ☐ On-Demand ☐ Data Transformation Service (DTS) ☐ No
2.	Does this application use MS Distributed Transaction Coordinator (MSDTC)? **(Please check one)** ☐ Yes ☐ No

Technical Diagrams

Please provide the following technical diagrams including the items specified below:

1. Proposed Architecture Diagram if different from current ITSD production environment (indicate hardware, OS, and software). An example is depicted in Figure A.1.0 below.
   a. For ITSD HASS Unit 1 note that this is QA/QC Item 2.4
2. Application Logical and Data Flow Diagram. An example is located in Figure A.2.0 below.
   a. For ITSD HASS Unit 1 note that this is QA/QC Item 4.2

Figure A.1.0 Physical Production Infrastructure Design

Example: Physical Production Infrastructure/Server Design/Implementation (Please include hardware, software, version and models).

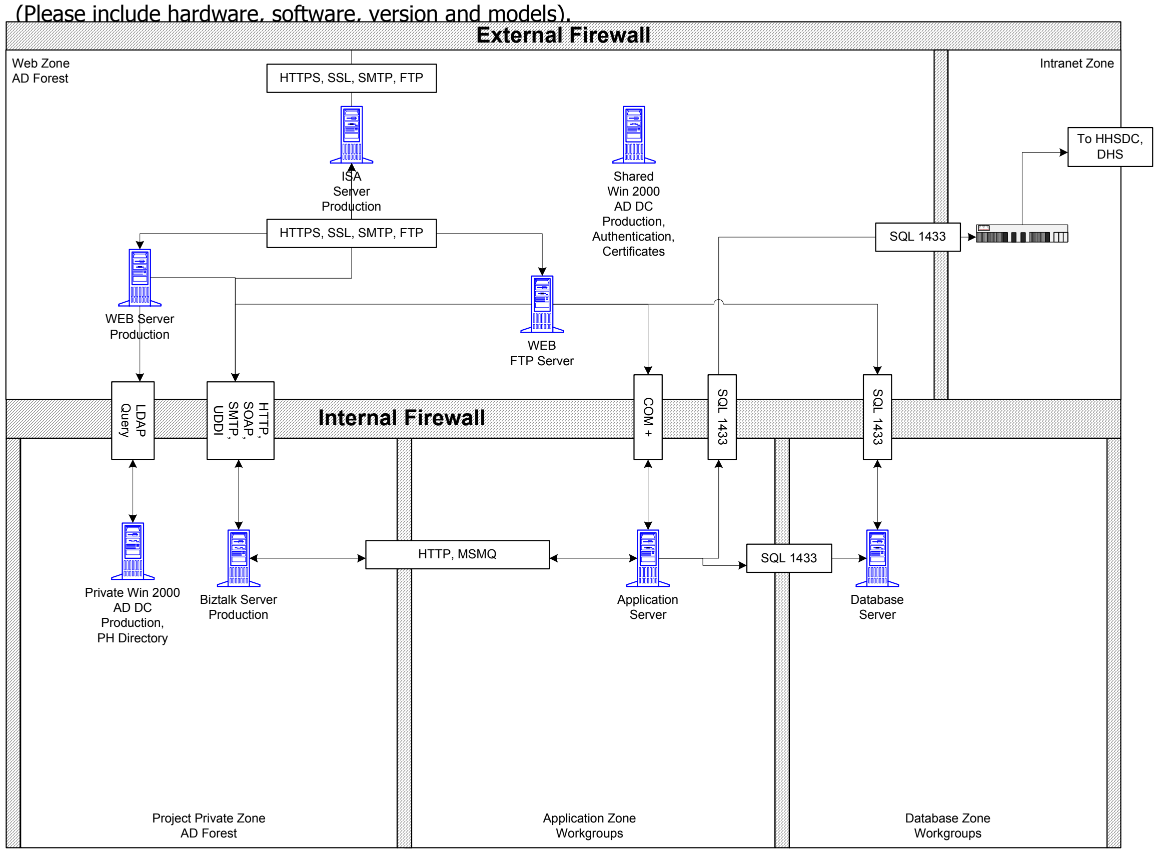
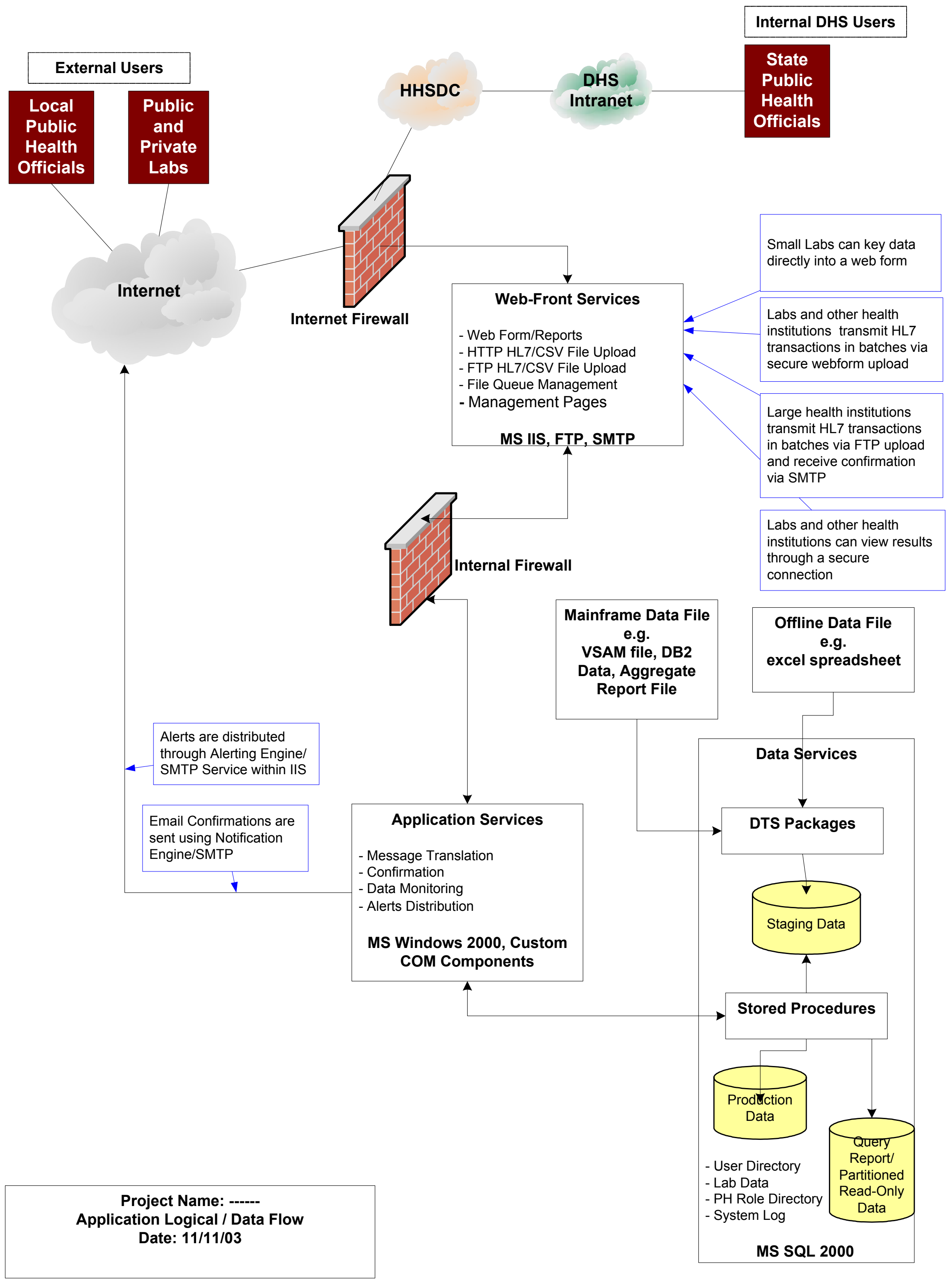


Figure A.2.0 Application Logical and Data Flow Sample Diagram


External Users
Local Public Health Officials
Public and Private Labs
HHSDC
DHS Intranet
Internal DHS Users
State Public Health Officials
Internet
Internet Firewall
Web-Front Services
- Web Form/Reports
- HTTP HL7/CSV File Upload
- FTP HL7/CSV File Upload
- File Queue Management
- Management Pages
MS IIS, FTP, SMTP
Small Labs can key data directly into a web form
Labs and other health institutions transmit HL7 transactions in batches via secure webform upload
Large health institutions transmit HL7 transactions in batches via FTP upload and receive confirmation via SMTP
Labs and other health institutions can view results through a secure connection
Internal Firewall
Mainframe Data File e.g. VSAM file, DB2 Data, Aggregate Report File
Offline Data File e.g. excel spreadsheet
Data Services
DTS Packages
Staging Data
Stored Procedures
Production Data
Query Report/ Partitioned Read-Only Data
- User Directory
- Lab Data
- PH Role Directory
- System Log
MS SQL 2000
Alerts are distributed through Alerting Engine/ SMTP Service within IIS
Email Confirmations are sent using Notification Engine/SMTP
Application Services
- Message Translation
- Confirmation
- Data Monitoring
- Alerts Distribution
MS Windows 2000, Custom COM Components
Project Name: ------
Application Logical / Data Flow
Date: 11/11/03

Systems Documentation

Please provide the following systems documentation:

1. Copy of the approved FSR or IFSR or Conceptual Paper.
    a. For ITSD HASS Unit 1 note that this is QA/QC Item 2.6
2. For ITSD HASS Unit 1 - Storyboard for your application
    a. For ITSD HASS Unit 1 note that this is QA/QC Item 3.1, 3.
3. Methods for obtaining and exposing data in this system including:
    a. Diagram
        i. Description of each of those methods, identifying:
            1. Where the users of the data enter the system
            2. The number of users accessing the system from each interface
            3. The number or volume of data being transacted including:
                a. Entry
                b. Automated Transfer
                c. Reporting
    b. Data retention policy for the system. For example, a 10-year history of data will be stored on SQL tables. As data from the current year is added to the tables, data from the 10th prior year will be achieved or deleted.
        i. How long will the data be kept, including:
            1. Staging Data
                a. Archive Data
                b. Purging Data
                c. Data transferring to other source
4. Document your proposed development environment:
    a. If you will be using the ITSD enterprise supported resources, please provide the following:

Hardware	**Software (include versions)**
Desktop	
Database Server	
Web Server	
Application Server	
Code Libraries or Non-Standard Components Proposed	

    b. Otherwise, please provide the following:

Hardware	**Software (include versions)**
Desktop	
Code Libraries or Non-Standard Components Proposed	

Appendix C Desktop Standards

The IT equipment within the CDHS is provided to support the department's goal of improving the health of all Californians. The costs of the resources allocated to design, implement, and maintain this equipment, must be minimized to ensure that spending on health care programs and services are maximized.

The CDHS IT vision is to provide secure networked personal computers with standard software that allow all users to perform work related functions such as e-mail, report creation, presentations, spreadsheets, etc. The content and links from this page outline the procurement, use, support, and retirement processes associated with the life cycle of these IT resources. Industry research and deployment history is available as well.

Please reference the most current Desktop Standards by visiting the CDHS Intranet site: http://itsd.int.dhs.ca.gov/ei/standards/.

Appendix D Server Standards

Refer to INFORMATION SECURITY STANDARDS ISO STANDARD NO. S17 CDHS SERVER STANDARDS located at : http://itsd.int.dhs.ca.gov/ei/standards/.

Appendix E Department of Health Services Research Center (CDHSRC) Architecture & Design

The CDHSRC is available for information technology research purposes. Please visit http://itsd.int.dhs.ca.gov/ei/dhsrc/ for the most current information.